# 가우시안 과정 분류를 위한 극단치에 강인한 학습 알고리즘

김현철
연세대학교
grass@postech.ac.kr

Zoubin Ghahramani
University of Cambridge
zoubin@eng.cam.ac.uk

# Outlier Robust Learning Algorithm for Gaussian Process Classification

Hyun-Chul Kim
Yonsei University

Zoubin Ghahramani
University of Cambridge

**Abstract**

Gaussian process classifiers (GPCs) are fully statistical kernel classification models which have a latent function with Gaussian process prior. Recently, EP approximation method has been proposed to infer the posterior over the latent function. It can have a special hyperparameter which can treat outliers potentially. In this paper, we propose the outlier robust algorithm which alternates EP and the hyperparameter updating until convergence. We also show its usefulness with the 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 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.

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 show experimental results on both synthetic and real data sets. In section 6, 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 [10] for a review of GPs). Assume that this GP prior is parameterized by $\Theta$ which we will call the hyperparameters. We can write the probability of interest given $\Theta$ as:

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

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.

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

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

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

The first term in Eq 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

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

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

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

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] and [5].

### 3. EP for GPC

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

$$p(y_i|f_i) = \epsilon + (1 - 2\epsilon) H(y_i f_i), \quad (6)$$

where $H(x) = 1$ if $x > 0$, and otherwise 0.

The hyperparameter, $\epsilon$ in Eq 6 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 $q(\mathbf{f}) \sim \mathcal{N}(\mathbf{m}_\mathbf{f}, \mathbf{V}_\mathbf{f})$, 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

$$C_{ij} = c(\mathbf{x}_i, \mathbf{x}_j) = v_0 \exp\{-\frac{1}{2}\sum_{m=1}^{d} l_m d_m(x_i^m, x_j^m)\} + v_1 + v_2\delta(i,j) \quad (7)$$

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

$$d_m(x_i^m, x_j^m) = \begin{cases} (x_i^m - x_j^m)^2 & \text{if } x^m \text{ is continous;} \\ 1 - \delta(x_i^m, x_j^m) & \text{if } x^m \text{ is discrete,} \end{cases} \quad (8)$$

and $\delta(x_i^m, x_j^m)$ is a Kronecker delta function.

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]}:

$$\mathbf{A} = \mathrm{diag}(v_1, \ldots, v_n);\ h_i = E[f_i];\ h_i^{\setminus i} = E[f_i^{\setminus i}],$$

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 $i$th 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: $z = \frac{y_i h_i^{\setminus i}}{\sqrt{\lambda_i}}$; $Z_i = \epsilon + (1-2\epsilon)\mathrm{erf}(z)$ $\alpha_i = \frac{1}{\sqrt{\lambda_i}}\frac{(1-2\epsilon)\mathcal{N}(z;0,1)}{\epsilon + (1-2\epsilon)\mathrm{erf}(z)}$; $h_i = h_i^{\setminus i} + \lambda_i\alpha_i$, where $\mathrm{erf}(z)$ is a cumulative normal density function.

(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: $\mathbf{A} = (\mathbf{C}^{-1} + \mathbf{A}^{-1})^{-1}$; 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:

$$q(\mathbf{f}) \sim \mathcal{N}(\tilde{\mathbf{C}}\boldsymbol{\alpha}, \mathbf{A}), \quad (9)$$

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:

$$p(Y|X,\Theta) \approx \frac{|\Lambda|^{1/2}}{|\mathbf{C}+\Lambda|^{1/2}}\exp(B/2)\prod_{i=1}^{n} s_i \quad (10),$$

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 Eq 10 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 10. In the following section, we derive the algorithm to find the hyperparameters automatically based not on Eq 10 but a variational lower bound of the evidence [8]. Classification of a new data point $\tilde{\mathbf{x}}$ can be done according to

$$\underset{\tilde{y}}{\mathrm{argmax}} p(\tilde{y}|\tilde{\mathbf{x}}) = \mathrm{sgn}(E[\tilde{f}]) = \mathrm{sgn}(\textstyle\sum_{i=1}^{n}\alpha_i y_i c(\mathbf{x}_i, \tilde{\mathbf{x}}))$$

## 4. Outlier Robust Learning Algorithm

We derive an algorithm that learns the labelling error hyperparameter, $\epsilon$, 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 $\epsilon$ and other model hyperparameters to maximize the evidence $p(Y|X,\epsilon)$. We put

$$\Theta = \Theta_{cov} \cup \{\epsilon\},$$

and $\Theta_{cov} = \{v_0, v_1, v_2\} \cup \{l_p | p = 1, 2, \ldots, d\}$.

The following procedure is iterated until convergence after initializing $\epsilon$ with a small value:

1. EP iterations are performed given the hyperparameter $\epsilon$ (see section 3). As a result of EP, the posterior $p(\mathbf{f}|D,\Theta)$ is approximated as a Gaussian density

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

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:

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

$$\log p(Y|X,\Theta) \geq \int q(\mathbf{f}) \log \frac{p(Y|\mathbf{f},\epsilon)p(\mathbf{f}|X,\Theta_{cov})}{q(\mathbf{f})} d\mathbf{f}$$

The only term in the lower bound that depends on $\epsilon$ is $F_\epsilon \stackrel{\text{def}}{=} \int q(\mathbf{f}) \log p(Y|\mathbf{f},\epsilon)\, d\mathbf{f}$, so we optimize $F_\epsilon$ with respect to $\epsilon$.

$$F_\epsilon = \sum_i \int q(f_i) \log p(y_i|f_i,\epsilon)\, df_i$$
$$= \sum_i \int q(f_i) \log(\epsilon + (1-2\epsilon)H(y_i f_i))\, df_i$$

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,

$$F_\epsilon = \sum_i (1-\omega_i) \log \epsilon + \omega_i \log(1-\epsilon)$$

Differentiating with respect to $\epsilon$ and solving we get: $\epsilon^* = \frac{1}{n}\sum_i (1-\omega_i)$

Although we have focused on learning the outlier model $\epsilon$, in step 2, the other hyperparameters of the covariance function $\Theta_{cov}$ 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. Simulation Results

We applied the proposed algorithm to a real world data set. We used New Thyroid data set which is from the UCI Machine Learning Repository[2]. New Thyroid data set has 5 continous variables, 3 (reduced to 2) classes, and 215 data points. New Thyroid daa set originally had three classes, "normal", "hyper" and "hypo", but we created a binary classification problem by grouping hyper and hypo into "not normal". Among 215 data points, 194 and 21 data points are selected as a training set and a test set, respectively.

We made outliers in a training set artificially with rates 0, 0.52, 1.55, 2.58, 3.61, 4.64 %. We applied EP with $\epsilon = 0$ and the proposed algorithm. Table 1 shows approximate log evidences[3], and test set error rates for each of algorithms and each of labeling error rates. In Table 1, 'lab. err.', 'for.', 'prop.' represent artificial labeling error rates, the former algorithm (EP with $\epsilon = 0$), the proposed algorithm. It shows EP with $\epsilon = 0$ does not work when the training set has outliers, but the proposed algorithm works robustly with outliers. EP with $\epsilon = 0$ worked quite badly especially in the cases that the labeling error is high. The proposed algorithm outperformed EP with $\epsilon = 0$ in senses of log evidence and test error. Interestingly, the labeling errors $\epsilon$ estimated from the proposed algorithm exactly matched the actual labeling errors. We tried 0.01, 0.02 and 0.03 as initial values of $\epsilon$ in the proposed algorithm, and got the same result.

| lab. err. (%) | $\log p(D)$ | | test err. (%) | |
|---|---|---|---|---|
| | for. | prop. | for. | prop. |
| 0 | -30.32 | -30.32 | 4.76 | 4.76 |
| 0.52 | -51.62 | -36.46 | 14.29 | 4.76 |
| 1.55 | -62.85 | -45.20 | 9.52 | 4.76 |
| 2.58 | -112.0 | -52.13 | 71.43 | 4.76 |
| 3.61 | -133.9 | -58.90 | 71.43 | 4.76 |
| 4.64 | -144.3 | -65.06 | 71.43 | 4.76 |

**Table 1 Comparison of the former and proposed algorithms**

## 6. 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 an outlier model is better than a normal model when there are outliers in data sets.

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.

[2] Available from http://www.ics.uci.edu/~mlearn/MPRepository.html

[3] The approximate log evidence was obtained from Eq 10 and means the model is better as its log evidence is higher.

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

**References**

[1] C. K. I. Williams and D. Barber, ""Bayesian Classification with Gaussian Processes," IEEE Transactions on PAMI, 20, pp. 1342-1351, 1998

[2] M. Gibbs and D. J. C. MacKay, "Variational Gaussian Process Classifiers," IEEE Transactions on NN, 11(6), p. 1458, 2000

[3] R. Neal, "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] M. Opper and O. Winther, "Gaussian processes for classification: Mean field algorithms," Neural Computation, 12, pp. 2655-2684, 2000

[5] T. Minka, "A family of algorithms for approximate Bayesian inference," PhD thesis, MIT, 2001

[6] Christopher K. I. Williams and Carl Edward Rasmussen, "Gaussian Processes for Regression," NIPS 8, MIT Press, 1995

[7] M. Seeger and N. Lawrence and R. Herbrich, "Sparse Representation for Gaussian Process Models," NIPS 15, MIT Press , 2002

[8] M. I. Jordan and Z. Ghahramani and T. S. Jaakkola and L. K. Saul, "An introduction to variational methods for graphical models," Machine Learning, 37, pp. 183-233, 1999

[9] R. Neal and G. Hinton, "A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants," Learning in Graphical Models, 1998

[10] H.-C. Kim and Z. Ghahramani, "The EM-EP algorithm for Gaussian process classification," Proceedings of the Workshop on Probabilistic Graphical Models for Classification (ECML), 2003