Homework: Gaussian Process Classification

Deadline: May 29 2022, 23:59*

1 Introduction

In this assignment, you will design a Gaussian Process Classifier on a image data set, a set of scanned handwritten digit images Optical character recognition (OCR) is the task of extracting text from image sources. The dataset on which you will run your classifiers is a collection of handwritten numerical digits (0-9). This is a very commercially useful technology, similar to the technique used by the post office to route mail by zip codes. There are systems that can perform with over 99% classification accuracy.

2 Project Instruction

The most part of this section is coming from the chapter 3 of the textbook [1]. You can download the pdf file from `http://www.gaussianprocess.org/gpml/`, and you are encouraged to read the book for better understanding this homework.

2.1 Multi-class Laplace Approximation

We first introduce the vector of latent function values at all n training points and for all C classes

$$\boldsymbol{a} = (a_1^1, ..., a_n^1, a_1^2, ..., a_n^2, ..., a_1^C, ..., a_n^C)^\top. \tag{1}$$

Thus $\boldsymbol{a}$ has length Cn. In the following we will generally refer to quantities pertaining to a particular class with superscript c, and a particular case by subscript i (as usual); thus e.g. the vector of C latents for a particular case is $\boldsymbol{a}_i$. However, as an exception, vectors or matrices formed from the covariance function for class c will have a subscript c. The prior over $\boldsymbol{a}$ has the form $\boldsymbol{a} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$. As we have assumed that the C latent processes are uncorrelated, the covariance matrix $\boldsymbol{K}$ is block diagonal in the matrices $\boldsymbol{K}_1, ..., \boldsymbol{K}_C$. Each individual matrix $\boldsymbol{K}_c$ expresses the correlations of the latent function values within the class c. Note that the covariance functions pertaining to the different classes can be different. Let $\boldsymbol{t}$ be a vector of the same length as $\boldsymbol{a}$ which for each $i = 1, ...n$ has an entry of 1 for the class which is the label for example i and 0 for the other $C-1$ entries.

Let π_i^c denote output of the softmax at training point i, i.e.

$$p(t_i^c|\boldsymbol{a}_i) = \pi_i^c = \frac{\exp(a_i^c)}{\sum_{c'} \exp(a_i^{c'})}. \tag{2}$$

Then $\boldsymbol{\pi}$ is a vector of the same length as $\boldsymbol{a}$ with entries π_i^c.

As we've seen from the class, the log-posterior for the binary classification case was:

$$\begin{aligned}\log p(\boldsymbol{a}|\boldsymbol{t}) &= \log p(\boldsymbol{a}) + \log p(\boldsymbol{t}|\boldsymbol{a}) + const \\ &= -\frac{1}{2}\boldsymbol{a}^\top \boldsymbol{K}^{-1}\boldsymbol{a} - \frac{1}{2}\log|\boldsymbol{K}| + \boldsymbol{t}^\top \boldsymbol{a} - \sum_{i=1}^{n} \log(1 + e^{a_i}) + const. \end{aligned} \tag{3}$$

The multi-class analogue of it is:

$$\log p(\boldsymbol{a}|\boldsymbol{t}) = -\frac{1}{2}\boldsymbol{a}^\top \boldsymbol{K}^{-1}\boldsymbol{a} - \frac{1}{2}\log|\boldsymbol{K}| + \boldsymbol{t}^\top \boldsymbol{a} - \sum_{i=1}^{n} \log\left(\sum_{c=1}^{C} \exp a_i^c\right) + const. \tag{4}$$

*You are encouraged to ask questions about this homework on icampus Q&A board. Please do not write your code on the board.

As in the binary case, we seek the MAP value $\boldsymbol{a}^*$ of $p(\boldsymbol{a}|\boldsymbol{t})$. By differentiating eq. (4) w.r.t. $\boldsymbol{a}$ we obtain

$$\nabla \log p(\boldsymbol{a}|\boldsymbol{t}) = \boldsymbol{t} - \boldsymbol{\pi} - \boldsymbol{K}^{-1}\boldsymbol{a}. \tag{5}$$

Thus at maximum we have $\boldsymbol{a}^* = \boldsymbol{K}(\boldsymbol{t} - \boldsymbol{\pi}^*)$. Differentiating again, and using

$$-\frac{\partial^2}{\partial a_i^c \partial a_i^{c'}} \log \sum_j \exp(a_i^j) = \pi_i^c \pi_i^{c'} - \pi_i^c \delta_{cc'}, \tag{6}$$

we obtain

$$\nabla^2 \log p(\boldsymbol{a}|\boldsymbol{t}) = -\boldsymbol{W} - \boldsymbol{K}^{-1}, \tag{7}$$
$$\boldsymbol{W} \triangleq \mathrm{diag}(\boldsymbol{\pi}) - \boldsymbol{\Pi}\boldsymbol{\Pi}^\top, \tag{8}$$

where $\boldsymbol{\Pi}$ is a $Cn \times n$ matrix obtained by stacking vertically the diagonal matrices diag$(\boldsymbol{\pi}^c)$, and $\boldsymbol{\pi}^c$ is the subvector of $\boldsymbol{\pi}$ pertaining to class c. As in the binary case notice that -$\nabla^2 \log p(\boldsymbol{a}|\boldsymbol{t})$ is positive definite, thus $\log p(\boldsymbol{a}|\boldsymbol{t})$ is concave and the maximum is unique.

As in the binary case we use Newton's method to search for the mode of $\log p(\boldsymbol{a}|\boldsymbol{t})$, which is $\boldsymbol{a}^*$, giving

$$\boldsymbol{a}^{new} = (\boldsymbol{K}^{-1} + \boldsymbol{W})^{-1}(\boldsymbol{W}\boldsymbol{a} + \boldsymbol{t} - \boldsymbol{\pi}). \tag{9}$$

This update if coded naively would take $\mathcal{O}(C^3n^3)$ as matrices of size Cn have to be inverted. However, as described in next subsection, we can utilize the structure of $\boldsymbol{W}$ to bring down the computational load to $\mathcal{O}(Cn^3)$.

The Laplace approximation gives us a Gaussian approximation $q(\boldsymbol{a}|\boldsymbol{t})$ to the posterior $p(\boldsymbol{a}|\boldsymbol{t})$. To make predictions at a test point $\boldsymbol{x}_{n+1}$ we need to compute the posterior distribution $q(\boldsymbol{a}_{n+1}|\boldsymbol{t})$ where $\boldsymbol{a}_{n+1} = (a_{n+1}^1, ...a_{n+1}^C)^\top$.

In general we have

$$q(\boldsymbol{a}_{n+1}|\boldsymbol{t}) = \int p(\boldsymbol{a}_{n+1}|\boldsymbol{a}) q(\boldsymbol{a}|\boldsymbol{t}) d\boldsymbol{a}. \tag{10}$$

As $p(\boldsymbol{a}_{n+1}|\boldsymbol{a})$ and $q(\boldsymbol{a}|\boldsymbol{t})$ are both Gaussian, $q(\boldsymbol{a}_{n+1}|\boldsymbol{t})$ will also be Gaussian and we need only compute its mean and covariance. The predictive mean for class c is given by

$$\mathbb{E}_q[a_{n+1}^c|\boldsymbol{t}] = \mathbf{k}_{c,n+1}^\top \boldsymbol{K}_c^{-1} \boldsymbol{a}^{*,c} = \mathbf{k}_{c,n+1}^\top (\boldsymbol{t}^c - \boldsymbol{\pi}^{*,c}), \tag{11}$$

where $\mathbf{k}_{c,n+1}$ is the vector of covariances between the test point and each of the training points for the cth covariance function, and $\boldsymbol{a}^{*,c}$ is the subvector of $\boldsymbol{a}^*$ pertaining to the class c. The last equality comes from eq. (5) at the maximum $\boldsymbol{a}^*$. Note the close correspondence to the binary case's predictive mean introduced at the class. This can be put into a vector form $\mathbb{E}_q[\boldsymbol{a}_{n+1}|\boldsymbol{t}] = \boldsymbol{Q}_{n+1}^\top(\boldsymbol{t} - \boldsymbol{\pi}^*)$ by defining the $Cn \times C$ matrix

$$\boldsymbol{Q}_{n+1} = \begin{pmatrix} \mathbf{k}_{1,n+1} & \mathbf{0} & ... & \mathbf{0} \\ \mathbf{0} & \mathbf{k}_{2,n+1} & ... & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & ... & \mathbf{k}_{C,n+1} \end{pmatrix}. \tag{12}$$

Using a similar argument, we obtain

$$\begin{aligned} Var_q[\boldsymbol{a}_{n+1}|\boldsymbol{t}] &= \Sigma + \boldsymbol{Q}_{n+1}^\top \boldsymbol{K}^{-1}(\boldsymbol{K}^{-1} + \boldsymbol{W})^{-1}\boldsymbol{K}^{-1}\boldsymbol{Q}_{n+1} \\ &= \mathrm{diag}(\mathbf{k}_{n+1,n+1}) - \boldsymbol{Q}_{n+1}^\top(\boldsymbol{K} + \boldsymbol{W}^{-1})\boldsymbol{Q}_{n+1}, \end{aligned} \tag{13}$$

where Σ is a diagonal $C \times C$ matrix with $\Sigma_{cc} = k_{c,n+1,n+1} - \mathbf{k}_{c,n+1}^\top \boldsymbol{K}_c^{-1} \mathbf{k}_{c,n+1}$, and $\mathbf{k}_{n+1,n+1}$ is a vector of covariances, whose c'th element is $k_{c,n+1,n+1}$.

We now need to consider the predictive distribution $p(\boldsymbol{\pi}_{n+1}|\boldsymbol{t})$ which is obtained by softmaxing the Gaussian $q(\boldsymbol{a}_{n+1}|\boldsymbol{t})$. Unlike the binary case, there is no obvious approximation method of doing this. One simple way to estimate the mean prediction $\mathbb{E}_q[\boldsymbol{\pi}_{n+1}|\boldsymbol{t}]$ is to draw samples from the Gaussian $q(\boldsymbol{a}_{n+1}|\boldsymbol{t})$, softmax them and then average.

The Laplace approximation to the marginal likelihood can be obtained in the same way as for the binary case, yielding

$$\begin{aligned}\log p(\boldsymbol{t}|\boldsymbol{\theta}) &\approx \log q(\boldsymbol{t}|\boldsymbol{\theta}) \\ &= \boldsymbol{t}^\top \boldsymbol{a}^* - \sum_{i=1}^{n} \log \left(\sum_{c=1}^{C} \exp a_i^{*,c} \right) \\ &\quad - \frac{1}{2}\boldsymbol{a}^{*\top}\boldsymbol{K}^{-1}\boldsymbol{a}^* - \frac{1}{2}\log |\boldsymbol{I}_{Cn} + \boldsymbol{W}^{1/2}\boldsymbol{K}\boldsymbol{W}^{1/2}| \end{aligned} \tag{14}$$

As the inversion of $\boldsymbol{K}^{-1} + \boldsymbol{W}$, the determinant term can be computed efficiently by exploiting the structure of $\boldsymbol{W}$, see next subsection.

2.2 Efficient Computation of Inverse & Determinant

The newton iteration from eq. (9) requires the inversion of $\boldsymbol{K}^{-1} + \boldsymbol{W}$, which we first re-write as

$$(\boldsymbol{K}^{-1} + \boldsymbol{W})^{-1} = \boldsymbol{K} - \boldsymbol{K}(\boldsymbol{K} + \boldsymbol{W}^{-1})^{-1}\boldsymbol{K}, \tag{15}$$

using the matrix inversion lemma. In the following the inversion of the above matrix $\boldsymbol{K} + \boldsymbol{W}^{-1}$ is our main concern. First, however, we apply the matrix inversion lemma to the $\boldsymbol{W}$ matrix:

$$\begin{aligned}\boldsymbol{W}^{-1} = (\boldsymbol{D} - \boldsymbol{\Pi}\boldsymbol{\Pi}^\top)^{-1} &= \boldsymbol{D}^{-1} - \boldsymbol{R}(\boldsymbol{R}^\top\boldsymbol{D}\boldsymbol{R} - \boldsymbol{I})^{-1}\boldsymbol{R}^\top \\ &= \boldsymbol{D}^{-1} - \boldsymbol{R}\boldsymbol{O}^{-1}\boldsymbol{R}^\top \end{aligned} \tag{16}$$

where $\boldsymbol{D} = \mathrm{diag}(\boldsymbol{\pi}), \boldsymbol{R} = \boldsymbol{D}^{-1}\boldsymbol{\Pi}$ is a $Cn \times n$ matrix of stacked $\boldsymbol{I}_n$ unit matrices, we use the fact that $\boldsymbol{\pi}$ normalizes over classes: $\boldsymbol{R}^\top\boldsymbol{D}\boldsymbol{R} = \sum_c \boldsymbol{D}_c = \boldsymbol{I}_n$ and $\boldsymbol{O}$ is the zero matrix. Introducing the above in $\boldsymbol{K} + \boldsymbol{W}^{-1}$ and applying the matrix inversion lemma again we have

$$\begin{aligned}(\boldsymbol{K} + \boldsymbol{W}^{-1})^{-1} &= (\boldsymbol{K} + \boldsymbol{D}^{-1} - \boldsymbol{R}\boldsymbol{O}^{-1}\boldsymbol{R})^{-1} \\ &= \boldsymbol{E} - \boldsymbol{E}\boldsymbol{R}(\boldsymbol{O} + \boldsymbol{R}^\top\boldsymbol{E}\boldsymbol{R})^{-1}\boldsymbol{R}^\top\boldsymbol{E} \\ &= \boldsymbol{E} - \boldsymbol{E}\boldsymbol{R}(\sum_c \boldsymbol{E}_c)^{-1}\boldsymbol{R}^\top\boldsymbol{E}. \end{aligned} \tag{17}$$

where $\boldsymbol{E} = (\boldsymbol{K} + \boldsymbol{D}^{-1})^{-1} = \boldsymbol{D}^{1/2}(\boldsymbol{I} + \boldsymbol{D}^{1/2}\boldsymbol{K}\boldsymbol{D}^{1/2})^{-1}\boldsymbol{D}^{1/2}$ is a block diagonal matrix and $\boldsymbol{R}^\top\boldsymbol{E}\boldsymbol{R} = \sum_c \boldsymbol{E}_c$. The Newton iterations can now be computed by inserting eq. (15) and eq. (17) instead.

Similarly, determinant can be computed in same manner applying the determinant lemma:

$$\log |\boldsymbol{I}_{Cn} + \boldsymbol{W}^{1/2}\boldsymbol{K}\boldsymbol{W}^{1/2}| = \log |\boldsymbol{W}^{-1} + \boldsymbol{K}| + \log |\boldsymbol{W}| \tag{18}$$

$$\log |\boldsymbol{W}| = \log |\boldsymbol{D} - \boldsymbol{\Pi}\boldsymbol{\Pi}^\top| = \log |\boldsymbol{O}| + \log |\boldsymbol{D}| \tag{19}$$

$$\begin{aligned}\log |\boldsymbol{W}^{-1} + \boldsymbol{K}| &= \log |\boldsymbol{K} + \boldsymbol{D}^{-1} - \boldsymbol{R}\boldsymbol{O}^{-1}\boldsymbol{R}^\top| \\ &= \log |\boldsymbol{R}^\top\boldsymbol{E}\boldsymbol{R}| + \log |\boldsymbol{K} + \boldsymbol{D}^{-1}| - \log |\boldsymbol{O}| \end{aligned} \tag{20}$$

$$\log |\boldsymbol{I}_{Cn} + \boldsymbol{W}^{1/2}\boldsymbol{K}\boldsymbol{W}^{1/2}| = \log |\sum_c \boldsymbol{E}_c| + \sum_c \log |\boldsymbol{I}_n + \boldsymbol{D}_c^{1/2}\boldsymbol{K}_c\boldsymbol{D}_c^{1/2}| \tag{21}$$

2.3 Model Selection for GP Classification

The performance of GP-based algorithms is heavily governed by the choice of the covariance function and the hyperparameter. The most widely used approach of choosing it is to compute the derivative of the marginal likelihood to optimize hyperparameter by gradient-based algorithms. Recall that the approximate log marginal likelihood was given as

$$\begin{aligned}\log q(\boldsymbol{t}|\boldsymbol{\theta}) &= \boldsymbol{t}^\top \boldsymbol{a}^* - \sum_{i=1}^{n} \log \left(\sum_{c=1}^{C} \exp a_i^{*,c} \right) \\ &\quad - \frac{1}{2}\boldsymbol{a}^{*\top}\boldsymbol{K}^{-1}\boldsymbol{a}^* - \frac{1}{2}\log |\boldsymbol{I}_{Cn} + \boldsymbol{W}^{1/2}\boldsymbol{K}\boldsymbol{W}^{1/2}| \end{aligned} \tag{22}$$

To this end we seek the partial derivatives of $\partial q(\boldsymbol{t}|\boldsymbol{\theta})/\partial\theta_j$. The covariance matrix $\boldsymbol{K}$ is a function of the hyperparameters, but $\boldsymbol{a}^*$ and therefore $\boldsymbol{W}$ are also implicitly functions of $\boldsymbol{\theta}$, since when $\boldsymbol{\theta}$ changes, the optimum of the posterior $\boldsymbol{a}^*$ also changes. Thus

$$\frac{\partial \log q(\boldsymbol{t}|\boldsymbol{\theta})}{\partial\theta_j} = \left.\frac{\partial \log q(\boldsymbol{t}|\boldsymbol{\theta})}{\partial\theta_j}\right|_{explicit} + \sum_{i=1}^{n} \frac{\partial \log q(\boldsymbol{t}|\boldsymbol{\theta})}{\partial \boldsymbol{a}_i^*}\frac{\partial \boldsymbol{a}_i^*}{\partial \boldsymbol{\theta}_j} \tag{23}$$

by the chain rule. The explicit term is then given by

$$\left.\frac{\partial \log q(\boldsymbol{t}|\boldsymbol{\theta})}{\partial\theta_j}\right|_{explicit} = -\frac{1}{2}\boldsymbol{a}^{*\top}\boldsymbol{K}^{-1}\frac{\partial \boldsymbol{K}}{\partial\theta_j}\boldsymbol{K}^{-1}\boldsymbol{a}^* - \frac{1}{2}tr\left((\boldsymbol{W}^{-1}+\boldsymbol{K})^{-1}\frac{\partial \boldsymbol{K}}{\partial\theta_j}\right). \tag{24}$$

When evaluating the remaining term, we utilize the fact that $\boldsymbol{a}^*$ is the maximum of the posterior, so that $\partial \log q(\boldsymbol{t}|\boldsymbol{a})/\partial\boldsymbol{a} = \boldsymbol{0}$ at $\boldsymbol{a} = \boldsymbol{a}^*$, where the log posterior is defined in eq. (4); thus the implicit derivatives of the two first terms vanish, leaving only

$$\begin{aligned} \frac{\partial \log q(\boldsymbol{t}|\boldsymbol{\theta})}{\partial \boldsymbol{a}_i^*} &= -\frac{1}{2}\frac{\partial \log|\boldsymbol{I}_{Cn} + \boldsymbol{W}^{1/2}\boldsymbol{K}\boldsymbol{W}^{1/2}|}{\partial \boldsymbol{a}_i^*} & (25)\\ &= -\frac{1}{2}tr\left((\boldsymbol{K}^{-1}+\boldsymbol{W})^{-1}\frac{\partial \boldsymbol{W}}{\partial \boldsymbol{a}_i^*}\right) & (26)\end{aligned}$$

In order to evaluate the derivative of $\partial\boldsymbol{a}^*/\partial\theta_j$, we use eq. (5) and the fact that gradient is zero to obtain $\boldsymbol{a}^* = \boldsymbol{K}(\boldsymbol{t}-\boldsymbol{\pi})$ and differentiate,

$$\frac{\partial \boldsymbol{a}^*}{\partial\theta_j} = \frac{\partial \boldsymbol{K}}{\partial\theta_j}(\boldsymbol{t}-\boldsymbol{\pi}) - \boldsymbol{K}\frac{\partial \boldsymbol{\pi}}{\partial \boldsymbol{a}^*}\frac{\partial \boldsymbol{a}^*}{\partial\theta_j} = (\boldsymbol{I}+\boldsymbol{K}\boldsymbol{W})^{-1}\frac{\partial \boldsymbol{K}}{\partial\theta_j}(\boldsymbol{t}-\boldsymbol{\pi}) \tag{27}$$

where we have used $\boldsymbol{W} = \partial\boldsymbol{\pi}/\partial\boldsymbol{a}^*$. Since the computation of this derivative is very complex, the function that calculates the derivative is given in case of the assignment.

2.4 Pseudo-code of the algorithm

Here, we attached the pseudo-code of the algorithm. You should implement your code based on this pseudo-code.

```
Algorithm 1: Mode-finding for multi-class Laplace GPC: findMode
  Input  : $\boldsymbol{X}$ (training data), $\boldsymbol{t}$ (0/1 targets), $\boldsymbol{\theta}$ (hyperparameter)
  Output: $\boldsymbol{a}^*$ (mode of posterior), $Z$ (approximate marginal likelihood)
  ------------------------------------------------------------
  Compute $\boldsymbol{K}_c$ using covariance function, with $\boldsymbol{X}$ and $\boldsymbol{\theta}$.
  $\boldsymbol{a} := \boldsymbol{0}$
  repeat
      Compute $\boldsymbol{b}, \boldsymbol{K}, logdet$ from calculateIntermediateValues
      $\boldsymbol{a} := \boldsymbol{K}\boldsymbol{b}$
  until convergence of the objective : $-\frac{1}{2}\boldsymbol{b}^\top\boldsymbol{a} + \boldsymbol{t}^\top\boldsymbol{a} - \sum_i \log(\sum_c \exp(a_c^i))$    ▷ Hint[a];
  $Z := objective - logdet$
```

[a]Difference between previous step of objective and current one is less than $1e-10$

Algorithm 2: Prediction for multi-class Laplace GPC: **calculatePredictiveDistribution**

```
Input   : $\boldsymbol{X}$ (training data), $\boldsymbol{t}$ (0/1 targets), $\boldsymbol{\theta}$ (hyperparameter), $\boldsymbol{a}^*$ (posterior mode), $\boldsymbol{x}_{n+1}$
          (test input)
Output: $\boldsymbol{\pi}_{n+1}$ (predicted class probability vector)
-------------------------------------------------------------------------------
Compute $\boldsymbol{K}_c, \mathbf{k}_{c,n+1}, k_{c,n+1,n+1}$ using covariance function, with $\boldsymbol{X}$, $\boldsymbol{x}_{n+1}$ and $\boldsymbol{\theta}$.
Compute $\boldsymbol{\pi}, \boldsymbol{E}_c, \boldsymbol{M}, \boldsymbol{R}$ from calculateIntermediateValues
Compute $\boldsymbol{R}_c$ by splitting $\boldsymbol{R}$
for $c := 1 \dots C$ do
    $\boldsymbol{\mu}^c_{n+1} = (\boldsymbol{t}^c - \boldsymbol{\pi}^c)^\top \mathbf{k}_{c,n+1}$          ▷ latent test mean from eq. (11)
    $\boldsymbol{f} := \boldsymbol{E}_c \mathbf{k}_{c,n+1}$
    $\boldsymbol{g} := \boldsymbol{E}_c(\boldsymbol{R}_c(\boldsymbol{M}^\top \backslash (\boldsymbol{M} \backslash (\boldsymbol{R}_c^\top \boldsymbol{f}))))$          ▷ Hint[a]
    for $c' := 1 \dots C$ do
        $\Sigma_{cc'} := \boldsymbol{g}^\top \mathbf{k}_{c',n+1}$
    end
    $\Sigma_{cc} := \Sigma_{cc} + k_{c,n+1,n+1} - \boldsymbol{f}^\top \mathbf{k}_{c,n+1}$          ▷ latent test cov from eq. (13)
end
$\boldsymbol{\pi}_{n+1} = \boldsymbol{0}$
for $i$:=1:$S$ do
    $\boldsymbol{a}_{n+1} \sim \mathcal{N}(\boldsymbol{\mu}_{n+1}, \Sigma)$
    $\boldsymbol{\pi}_{n+1} = \boldsymbol{\pi}_{n+1} + \exp(a^c_{n+1}) / \sum_{c'} \exp(a^{c'}_{n+1})$
end
$\boldsymbol{\pi}_{n+1} = \boldsymbol{\pi}_{n+1} / S$          ▷ MC estimate of prediction vector
```

[a]The meaning of $\boldsymbol{A} \backslash \boldsymbol{b}$ is the vector $\mathbf{x}$ which solves $\boldsymbol{A}\mathbf{x} = \boldsymbol{b}$. You can get more information on `http://www.gaussianprocess.org/gpml/chapters/RWA.pdf`.

Algorithm 3: Function that calculates the values needed: **calculateIntermediateValues**

```
Input   : $\boldsymbol{K}_c$ (covariance matrices), $\boldsymbol{t}$ (0/1 targets), $\boldsymbol{a}$ (mode of posterior)
-------------------------------------------------------------------------------
Compute $\boldsymbol{\pi}$ from $\boldsymbol{a}$ with eq. (2)
$\boldsymbol{K} = bkdiag(\boldsymbol{K}_c)$
$\boldsymbol{D} := bkdiag(\boldsymbol{\pi})$
$logdet := 0$
for $c$:=1...$C$ do
    $\boldsymbol{D}_c = diag(\boldsymbol{\pi}_c)$
    $\boldsymbol{L} := cholesky(\boldsymbol{I}_n + \boldsymbol{D}_c^{1/2} \boldsymbol{K}_c \boldsymbol{D}_c^{1/2})$
    $\boldsymbol{E}_c := \boldsymbol{D}_c^{1/2}(\boldsymbol{L}^\top \backslash (\boldsymbol{L} \backslash \boldsymbol{D}_c^{1/2}))$
    $logdet := logdet + \sum_i \log \boldsymbol{L}_{ii}$
end
$\boldsymbol{M} := cholesky(\sum_c \boldsymbol{E}_c)$          ▷ Hint[a]
$\boldsymbol{E} := bkdiag(\boldsymbol{E}_c)$
$logdet := logdet + \sum_i \log \boldsymbol{M}_{ii}$
Compute $\boldsymbol{\Pi}$ by defn. under eq. (8)
Set $\boldsymbol{R}$ by defn. under eq. (16)
$\boldsymbol{c} := (\boldsymbol{D} - \boldsymbol{\Pi}\boldsymbol{\Pi}^\top)\boldsymbol{a} + \boldsymbol{t} - \boldsymbol{\pi}$          ▷ $\boldsymbol{c} = \boldsymbol{W}\boldsymbol{a} + \boldsymbol{t} - \pi$ from eq. (9)
$\boldsymbol{d} := \boldsymbol{E}\boldsymbol{K}\boldsymbol{c}$
$\boldsymbol{b} := \boldsymbol{c} - \boldsymbol{d} + \boldsymbol{E}\boldsymbol{R}(\boldsymbol{M}^\top \backslash (\boldsymbol{M} \backslash (\boldsymbol{R}^\top \boldsymbol{d})))$          ▷ part of eq. (9) using (15) and (17)
```

[a]You can compute the $\boldsymbol{M} := cholesky(\sum_c \boldsymbol{E}_c + \epsilon \boldsymbol{I})$ where ϵ is small values such as $1e-6$ to avoid the computing errors. The grader uses this approach.

2.5 What to Do

You must fill the portion of **gpClassification.py** during the assignment. You will fill in the following functions for this assignment:

- *find_mode*
- *calculate_intermediate_values*
- *calculate_predictive_distribution*

, all of which the pseudo-code is given above.

2.6 Some Hints and Requirements

- The time complexity of GP classification is $O(n^3)$; try with small amount of data first.
- You can only use numpy and scipy modules. If you use/import any other libraries, you will get zero points.
- Some functions are predefined in the GaussianProcessMultiClassifier class. It would be better to use the methods.
- You are encouraged to ask questions about this homework on icampus Q&A board. Please do not write your code on the board since TAs can see your code in icampus.
- We also test the running time of your code. To avoid the congestion of submission and running the code, you would better to avoid to run the code at the deadline.
- You'd better to read the existing code carefully. You can use the existing methods such as softmax to make the complete code.
- For predicting new data, you don't need to update the hyperparameters. Please use self.hyp variable which is initialize at the beginning of the training time.
- You are encouraged to check the shape of the matrix and vectors carefully.

3 Submission and Evaluation

You need to submit the followings:

- gpClassification.py

You MUST compress the above folders with your own source code files into 'HW3_yourid.zip' (e.g.), HW3_2021123456.zip and submit the compressed file to icampus [2]. If you do not follow the above submission policy, you will get penalty.

We will be checking your code against other submissions in the class for logical redundancy. If you copy someone else's code and submit it with minor changes, we will know. These cheat detectors are quite hard to fool, so please don't try. We trust you all to submit your own work only. If you do, we will give F grade.

You are not alone! If you find yourself stuck on something, contact the TA for help. Please use icampus Q&A board. If you are in a struggle, other students are also the same. You can ask common questions, and you can find the solutions by searching. But please follow the communication rules in the first week's pdf file. If not, we might ignore your questions.

References

[1] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

[2] SKKU i-Campus. `https://icampus.skku.edu/`, 2022.