Boosting Gaussian Process Classification Performance via Optimal Kernel Selection

Demetrius Rowland

13 May 2019

Introduction

Gaussian Processes (GPs) are powerful tools for the estimation of functions from a finite set of ordered pairs $(x_1, y_1), ..., (x_n, y_n)$. They can be readily incorporated into a bayesian framework and they lend themselves to simple, albeit computationally demanding, calculations. However, as we shall soon see, their performance can be limited by the choice of kernel. Therefore, we seek methods to choose the kernel based on the problem at hand, and we will explore two criteria for selection: kernel selection via maximizing marginal likelihood and kernel selection via minimizing PAC generalization error bounds. The plan is as follows:

1. Introduce GP Regression and Classification
2. Introduce benchmark models for comparison
3. Test all models on a COVID-19 data set
4. Compute marginal likelihoods for the COVID-19 training data under several distinct kernels
5. Compute PAC learning bounds using the same kernels
6. Select the kernels according to the aforementioned criteria

Gaussian Processes

A **Gaussian Process** is a collection of random variables $\{f(x)\}_{x \in \mathcal{X}}$, for which any finite subcollection $\{f(x_1), ..., f(x_m)\}$, $m \in \mathbb{Z}^+$ is multivariate normal. It is completely specified by its mean function $m(x)$ and its covariance function $k(x, x')$. Starting here, we will refer to the covariance function as the **kernel** of the Gaussian Process.

Given a mean function m and kernel k, we can find the distribution of any finite subcollection $\{f(x_1), ..., f(x_m)\}$, $m \in \mathbb{Z}^+$ of a GP from the definition:

$$f = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_m) \end{bmatrix} \sim \text{Normal}\,(\begin{bmatrix} m(x_1) \\ \vdots \\ m(x_m) \end{bmatrix}, \; \begin{bmatrix} k(x_1,x_1) & \dots & k(x_1,x_m) \\ \vdots & \ddots & \vdots \\ k(x_m,x_1) & \dots & k(x_m,x_m) \end{bmatrix})$$

In our preliminary analysis of these data sets, we use a mean function

$$m(x) = 0 \qquad \text{(zero mean)}$$

and kernel

$$k(x,y) = \exp\,(-\frac{1}{2} \mid\mid x - y \mid\mid^2) \qquad \text{(squared exponential kernel)}$$

and we denote

$$\begin{aligned} f(x) &\sim \text{GP}\,(m(x),\; k(x,x')) \\ &= \text{GP}\,(0,\; \exp\,(-\frac{1}{2} \mid\mid x - x' \mid\mid^2)) \end{aligned}$$

Under this formalism, our objective is to perform classification.

We find it useful to describe regression using Gaussian Processes first, for reasons that will become apparent.

Our treatment follows Rasmussen and Williams [1].

Regression Using GPs

We define **regression** to be the task of estimating a continuous target variable $Y^\star$, given feature input $X^\star$, a feature training set X and training targets Y.

How can we perform regression using Gaussian Processes?

We assume that $Y, Y^\star$ were sampled from a Gaussian Process with the zero mean and the squared exponential kernel. Then we must have that

$$\begin{aligned} \begin{bmatrix} Y \\ Y^\star \end{bmatrix} &\sim \text{Normal}\,(\begin{bmatrix} m(X) \\ m(X^\star) \end{bmatrix}, \; \begin{bmatrix} k(X,X) & k(X,X^\star) \\ k(X^\star,X) & k(X^\star,X^\star) \end{bmatrix}) \\ &\sim \text{Normal}\,(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \; \begin{bmatrix} k(X,X) & k(X,X^\star) \\ k(X^\star,X) & k(X^\star,X^\star) \end{bmatrix}) \end{aligned}$$

We use a standard result about the multivariate normal distribution to condition $Y^\star$ on Y, and we obtain

$$Y^\star \mid Y \sim \text{Normal}\,(\mu_{Y^\star|Y}, \Sigma_{Y^\star|Y})$$

where

$$\begin{aligned}\mu_{Y^\star|Y} &= k(X^\star, X)\, k(X, X)^{-1}\, Y \\ \Sigma_{Y^\star|Y} &= k(X^\star, X^\star) - k(X^\star, X)\, k(X, X)^{-1}\, k(X, X^\star)\end{aligned}$$

We can now make predictions for $Y^\star$ by taking the posterior mean, sampling from the posterior distribution, or using the posterior distribution directly.

Classification Using GPs

We define **classification** to be the task of assigning exactly one of m discrete labels $Y^\star$ to a given feature input $X^\star$, a feature training set X and training labels Y. We first consider **binary classification** in which $m = 2$.

In order to perform binary classification we assume the model for $Y^\star$ has the form

$$Y^\star \mid f^\star = \begin{cases} 1 & \text{with probability } p = \sigma(f^\star) \\ 0 & \text{with probability } 1 - p \end{cases}$$

where the latent variable $f^\star$ is assumed to be a Gaussian Process evaluated at the points $X^\star$, and $\sigma(x) = \frac{1}{1+\exp(-x)}$.

Thus binary classification requires us to carry out two steps:

1. Evaluation of the latent posterior

   $$p(f^\star \mid X, Y, X^\star) = \int p(f^\star \mid X, X^\star, f)\, p(f \mid X, Y)\, df$$

   where f denotes the latent GP evaluated at the training points X.

2. Evaluation of the predictive probability

   $$p(Y^\star = 1 \mid X, Y, X^\star) = \int \sigma(f^\star)\, p(f^\star \mid X, Y, X^\star)\, df^\star$$

Notice that we have already found $p(f^\star \mid X, X^\star, f)$ in our analysis of GP Regression, so it was convenient that we introduced regression first.

However, observe that the density

$$p(f \mid X, Y) = \frac{p(Y \mid f)\, p(f \mid X)}{p(Y \mid X)}$$

requires us to compute the marginal density $p(Y \mid X)$, which is intractable. In the case of regression, we were fortunate that $p(Y \mid X)$ was normal, but the introduction of the sigmoid function for binary classification means that this is no longer the case.

What we can do to resolve this issue is introduce a Laplace Approximation

$$q(f \mid X, Y) = \text{Normal}\left(f \mid \widehat{f}, A^{-1}\right)$$

to the density $p(f \mid X, Y)$, where

$$\begin{aligned}
\widehat{f} &= \text{argmax}_f \; p(f \mid X, Y) \\
A &= -\frac{d^2}{df^2} \log p(f \mid X, Y) \mid_{f=\widehat{f}}
\end{aligned}$$

Under this approximation, the latent posterior for $f^\star$,

$$p(f^\star \mid X, Y, X^\star) = \int p(f^\star \mid X, X^\star, f) \, q(f \mid X, Y) \, df$$

can be computed exactly as

$$\begin{aligned}
p(f^\star \mid X, Y, X^\star) &= \text{Normal}\left(\mu_{f^\star \mid X, Y, X^\star}, \; \Sigma_{f^\star \mid X, Y, X^\star}\right) \\
\mu_{f^\star \mid X, Y, X^\star} &= k(X^\star, X) \, k(X, X)^{-1} \, \widehat{f} \\
\Sigma_{f^\star \mid X, Y, X^\star} &= k(X^\star, X^\star) - k(X^\star, X) \, A^{-1} \, k(X, X^\star)
\end{aligned}$$

as in the case of regression, and we can compute the predictive probability in step (2) via numerical integration techniques.

With all the appropriate theory established, we are now ready to perform binary classification using Gaussian Processes. How can we extend our work to include classification for $m > 2$ classes?

We choose to implement a One-Versus-Rest Classification scheme, detailed below in pseucode.

Algorithm 1: One Versus Rest GP Classification

```
input  : $X, Y, X^\star, m$, Binary GP Classifier
output: Class Estimates for Test Points $X^\star$
int n = len(Y);
for $c$ = 1, ..., $m$ do
    vector $Y_c$;
    for $i$ = 1, ..., $n$ do
        if $(Y)_i == c$ then
            $(Y_c)_i$= 1
        else
            $(Y_c)_i = 0$
        end
    end
    $\pi_c$ = Binary GPC $(X, Y_c, X^\star)$;
end
$Y_{\text{estimates}} = \text{argmax}_c \pi_c$;
return $Y_{estimates}$
```

Note that this does not give a distribution over classes for each test point, but a class estimate. However, we will find that this scheme is sufficient for the task at hand, and we are ready to use this multi-class GP classifier on our COVID-19 data.

Benchmark Models

In order to benchmark the performance of our model, we introduce three standard classifiers: Support Vector Machines, Neural Networks, and the Logistic Regression model.

A **Support Vector Machine** (SVM) is a binary linear classifier that maximizes the distance between the classifying hyperplane and the nearest training point. More formally, we specify that an SVM is the model

$$y = \begin{cases} 1 & w^T x + b \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where w, b are chosen so that

1. The distance between the hyperplane H to the nearest point x_0 is maximized.

2. Each training point is correctly classified, i.e.

$$(w^T X_i + b)(2Y_i - 1) \geq 0 \qquad \text{for } i = 1, ..., \text{num\_train}$$

This can be reduced to a constrained optimization, for which a Lagrangian and corresponding dual problem can be formulated. See Christianini and Shawe-Taylor [2] for further details.

A **Neural Network** is a sequence of computations of the form

$$a_i(W_i o_{i-1} + b_i) \quad \text{for i = 1, ..., n}$$

where n is used to denote the number of layers, a_i is called the activation function, W_i is called the weight matrix, and b_i is called the bias for layer i, and o_{i-1} is the output of layer $i-1$.

They are trained by minimizing some loss function L with respect to the parameters W_i, b_i given training set X and training targets Y. This can be done efficiently using gradient descent and backpropagation, for which the literature is extensive. See Aggarwal [3].

A **Logistic Regression** model is a model of the form

$$\mathbb{P}(y = 1 \mid w, b, x) = \sigma(w^T x + b)$$

where w, b are typically chosen so as to maximize the likelihood of the training labels Y given the training set X. Because of the introduction of the sigmoid function $\sigma(x)$, a closed form solution for w, b does not exist, so we resort to numerical methods.

Having given some background for these models, we proceed to compare them against our GP classifier.

Data

We test our GP on a data set consisting of the chest CT scans of patients that fall into exactly one of three categories:

1. Patients infected with COVID-19 (label 0)
2. Normal Patients (label 1)
3. Patients with Viral Pneumonia (label 2)

The dimension of each image is $1024 \times 1024 \times 3$, corresponding to 1024×1024 red, green, and blue values. We flatten each image to make a vector of dimension 3145728 for each image, and we collect the 2902 image vectors to form a training set $X \in \mathbb{R}^{2902 \times 3145728}$. We also collect the labels for each image and form training labels $Y \in \mathbb{R}^{2902}$.

Note that the number of columns in X is too large for our models to handle efficiently. Therefore we implement dimension reduction and keep d = 1000 columns to obtain a matrix $X_{\text{reduced}} \in \mathbb{R}^{2902\times 1000}$, via Principal Component Analysis, for which the details can be found in the Appendix.

We then randomly permute the rows of the identity matrix $I \in \mathbb{R}^{2902}$ to obtain a permutation matrix P, and we set $X = PX$, $Y = PY$. Finally, we use an 80/20 split for training and testing, to obtain a training set X, training labels Y, a test set $X^\star$, and test labels $Y^\star$.

Preliminary Results

Implementing our GP Classifier on the chest CT scans and comparing its performance against the three benchmark models produces the following table of results.

	GP Classifier	SVM	Neural Net	Logistic Regressor
Test Accuracy	.4974	.9466	.8830	.9053

where the test accuracy is defined to be the proportion of test points correctly classified by the model.

The results suggest that the GP classifier must be improved, since it is beaten by every other model, so we choose to modify the kernel of our GP classifier according to the two schemes suggested in the introduction.

Kernel Selection via Marginal Likelihood

It is common to implement bayesian model selection in order to distinguish between models. However, this requires the evaluation of a sequence of integrals, which can be tedious, especially because of the introduction of the sigmoid function in GP classification.

Therefore we choose instead to maximize the marginal likelihood

$$p(Y \mid X, \theta, M_i) = \int p(y \mid X, f, M_i)\, p(f \mid \theta, M_i)\, df$$

with respect to the hyperparameters θ, and the models M_i, which are in our case completely specified by our choice of kernel.

We choose, then, to fix four kernels (the constant kernel, the Matern kernel, the rational quadratic kernel, and the white kernel), and hence four models $M_1, ..., M_4$. We optimize the marginal likelihoods w.r.t. the hyperparameters under each model, and we select the model for which this marginal likelihood is greatest.

The results are included in the following table:

	Constant	Matern	Rational Quadratic	White
Log Marginal Likelihood	-1282	-1625	-603.5	-1611
Test Accuracy	.4492	.4974	.9398	.4819

Thus the kernel selected under this scheme is the Rational Quadratic, which has a test accuracy of .9398, so this classifier performs far better than its predecessor and does about as well as the Support Vector Machine. We now turn our attention to the second scheme, in which we select the model which minimizes the PAC generalization bounds.

Kernel Selection via PAC Learning Bounds

The **PAC Learning Model** is a framework for placing an upper bound ϵ on the generalization error of a learning algorithm, such that probability of the compliance with the bound ϵ is $1 - \delta$.

Here, we define the generalization error to be the risk $R(y, \widehat{y})$, where y is the true class label, $\widehat{y}$ is our estimate and the underlying loss function is

$$L(y, \widehat{y}) = \begin{cases} 1 & \widehat{y} \neq y \\ 0 & \text{otherwise} \end{cases}$$

Then according to Seeger [4], we claim that for any set $\mathcal{X}$, for any distribution D over $\mathcal{X} \times \{0, 1\}$, for any $0 \leq \delta \leq 1$, and for any probability measures p and q over the unknown function values $[f, f^\star]^T$, we can make the following bound for n i.i.d. samples drawn from D:

$$p_D(\mathrm{KL}(\widehat{R}\,(y, \widehat{y}) \;||\; R\,(y, \widehat{y})) \leq \frac{1}{n}\,(\mathrm{KL}\,(q \;||\; p) + \log\frac{n+1}{\delta})) \geq 1 - \delta$$

where $\widehat{R} = \frac{1}{n}\sum_{i=1}^{n} L(Y_i, \widehat{Y}_i)$ is the empirical risk and KL denotes the KL-divergence between two probability measures.

In the case of GP Laplace binary classification, we set p to be the prior over function values $[f, f^\star]^T$, q to be the posterior given training labels Y, and we seek to minimize the KL-divergence KL $(q \;||\; p)$, which is given in Rasmussen and Williams [1] in closed form:

$$\mathrm{KL}\,(q \;||\; p) = \frac{1}{2}\log|K| + \frac{1}{2}\log|A| + \frac{1}{2}\mathrm{tr}\,(A^{-1}(K^{-1} - A)) + \frac{1}{2}\widehat{f}^T K^{-1}\widehat{f}$$

where K is the covariance matrix obtained by evaluating the kernel at the training points X, $A = K^{-1} + W$, where W is the negative Hessian of the log likelihood $\log p(Y \mid f)$, and $\widehat{f}$ is the maximizing value of the posterior $p(f \mid X, Y)$.

With this established, we compute KL ($q \mid\mid p$) for each of the models and the select the model for which it is minimized. The results are given in the following table.

	Constant	Matern	Rational Quadratic	White
KL ($q \mid\mid p$)	.7643	4.784	2.173	13.04
Test Accuracy	.4492	.4974	.9398	.4819

So we select the Constant Kernel, which gives a test accuracy of .4492. This confirms Seeger's fears that using the PAC bounds for model selection would result in a poor choice, because the generalization error is often many times smaller than the bounds suggested by the PAC learning model. For future analysis, we will resort to maximizing the marginal likelihood, cross-validation, or bayesian model selection.

Appendix

Principal Component Analysis is a dimension reduction technique in which a set of row vectors $X \in \mathbb{R}^{n \times p}$ is projected onto a d-dimensional affine space with center $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and a basis U, which is chosen so as to minimize the sum of the squares of the distances between each point X_i and the projection of X_i onto the subspace.

Following Vidal [5], this can be accomplished by subtracting $\overline{X}$ from each row of the matrix X, to form a centered matrix W, and carrying out a singular value decomposition

$$W^T = U_W \Sigma_W V_W^T$$

Then U is given by the first d columns of U_W and the projections y_j are given by the top $d \times n$ submatrix of $\Sigma_W V_W^T$. In our work, we implement principal component analysis using scikit-learn.