Gaussian Processes Classification and its PAC-Bayes Generalization Error Bounds – CSE 291 Project Report

Chicheng Zhang

March 20, 2013

1 The PAC-Bayes Theorem and application in GP Classification

1.1 The PAC-Bayes Theorem

McAllester's PAC-Bayes theorem (strengthened by [4]) characterizes the convergence of a stochastic classifier's empirical error to its generalization error. Fixed one "prior" distribution $P(h)$ over hypothesis space $\mathcal{H}$, the theorem can hold for all "posterior" distribution $Q(h)$ over $\mathcal{H}$ simultaneously, so in practice we can find a data-dependent posterior distribution over $\mathcal{H}$ as the distribution of final stochastic classifier, and achieve data-dependent gap error bound $\frac{D(Q||P)+\ln\frac{n+1}{\delta}}{n}$. These generalization error bounds can be sharper than trivially using traditional uniform convergence to bound its performance.

Theorem 1. *For any data distribution over $\mathcal{X}\times\{-1,+1\}$, fix a "prior" $P(h)$ over the hypothesis space $\mathcal{H}$, then with probability $1-\delta$ over the random draw of training examples $S=\{(x_1,y_1),\ldots,(x_n,y_n)\}$, we have for all "posterior" $Q(h)$:*

$$D_{Ber}(E_{h\sim Q}er_S(h)||E_{h\sim Q}er_D(h)) \leq \frac{D(Q||P)+\ln\frac{n+1}{\delta}}{n} \tag{1}$$

The PAC-Bayes theorem is the cornerstone of our testing the generalization error bounds. In the project, we assume each classifier $h(x)$ correspond to a set of underlying real valued functions $f(x)$, i.e. all $f(x)$ such that $h(x)=\text{sgn}(f(x))$. We assume the prior distribution over $\{f:X\to\mathbb{R}\}$ is a Gaussian process. Since we want to keep our calculation of the parameters of the bounds tractable, we concentrate on posterior processes which is also Gaussian, so that the relative entropy of these two processes proves to have a closed form. Since h is a deterministic function of f, log sum inequality implies the relative entropy of prior and posterior of h is bounded by that of the relative entropy of prior and posterior of f(aka chain rule of relative entropy, merging outcomes only decreases relative entropy), formally:

$$D(Q(h)||P(h)) \leq D(Q(f)||P(f))$$

Although PAC-Bayes depends only on the LHS, we upper bound it using the RHS, which has a nice closed form.

1.2 Generative Model and inference

Throughout the whole project, we assume the generative model as follows: first sample a function $f:X\to R$ from the underlying Gaussian process(defined by mean function $\mu(x)\equiv 0$, and covariance function $k(x,x')$ as a kernel function), then sample labels $y|x\sim p(y|f(x))$, which can be in probit noise model $p(y|f(x))=\Phi(y(\lambda f(x)+b))$ or logistic noise model $p(y|f(x))=\sigma(yf(x))=\frac{1}{1+\exp(-yf(x))}$. One view of this restricted to training data is:

(1) sample $f(x_1),\ldots,f(x_n)$ from the underlying Gaussian process, denoted as vector f_S.

(2.1) sample $y_i|f(x_i) \sim p(y|f(x_i))$, $\forall i = 1, 2, \ldots, n$.

(2.2) sample $f(x), x \notin \{x_1, \ldots, x_n\}$ from the conditional Gaussian Process given $x_1, \ldots, x_n$, whose finite dimensional distribution (abbrev. f.d.d.) is

$$f_Z|f_S \sim N(k(Z,S)k(S,S)^{-1}f_S, k(S,S) - k(Z,S)k(S,S)^{-1}k(S,Z))$$

In this view, the conditional independence can be easily seen:

$$(y_1, \ldots, y_n) \perp\!\!\!\perp \{f(x), x \notin \{x_1, \ldots, x_n\}\}|(f(x_1), \ldots, f(x_n))$$

So the posterior process can be broken to two parts: the first is purely the posterior distribution of f values on training data. (We use Gaussian approximation to keep tractability.) The second is the distribution of other data points' $f(x), x \notin \{x_1, \ldots, x_n\}$, conditioned on $f(x_1), \ldots, f(x_n)$, which is the same as the corresponding part of the prior, which cancels out in calculating the relative entropy. To see this, we note:

Prior process' f.d.d.(wlog, we always include f_S in f.d.d., i.e $S \subset Z$):

$$P(f_Z) = P(f_S) \times p(f_Z \backslash f_S|f_S)$$

Posterior process's f.d.d.:

$$P(f_Z|S) = P(f_S|S) \times p(f_Z \backslash f_S|f_S)$$

So the relative entropy of the two processes is essentially the relative entropy of their finite dimensional distribution constrained on the training set $x_1, \ldots, x_n$. So it can be evaluated in closed form; assume

$$P(f_S|S) = N(f_S|K\alpha_S, \Sigma_S) \tag{2}$$

where $K = (k(x_i, x_j))_{n \times n}$. Then:

$$D(Q||P) = \frac{1}{2}(\ln \Sigma_S^{-1} K + tr((\Sigma_S^{-1}K)^{-1}) + \alpha_S^T K \alpha_S - n)$$

which is the main term of the gap error bound in (1).

The classification rule of the GP is to produce a posterior distribution over the f values of the testing examples. Traditionally only the mean is valuable to make a prediction, while in selective classification problems the variance can also be taken into account, because combining mean and variance together can potentially give better characterization of whether the algorithm is sure to classify the testing example or not. Note that in traditional SVM based selective classification, the algorithm only care about the decision values and does not produce variances of examples at all.

For a testing point x_*, denote $f_* = f(x_*)$, $k_* = (k(x_*, x_1), \ldots, k(x_*, x_n))^T$, according to standard properties of Gaussian distribution:

$$P(f_*|f_S) = N(k_*^T K^{-1} f_S, k(x_*, x_*) - k_*^T K^{-1} k_*)$$

So the posterior distribution of f_* is:

$$P(f_*|S) = N(k_*^T \alpha_S, k(x_*, x_*) - k_*^T (K^{-1} - K^{-1}\Sigma_S K^{-1}) k_*) \tag{3}$$

In sum, the key of our approach to find a suitable approximation of the α_S and Σ_S in (2), then the generic result of (3) can be directly applied.

2 A first approximation: GP with logistic regression + Laplace approximation

In logistic regression, we assume noise model to be

$$P(y_i|f(x_i)) = \sigma(y_i f(x_i))$$

Then the log posteior can be represented as:

$$\ln(p(f_S|S)) = \sum_{i=1}^{n} \ln \sigma(y_i f(x_i)) - f_S^T K^{-1} f_S^T + const$$

The Laplace approximation essentially finds a mode of the posterior to be the mean, then relate the Hessian of the log-posterior at mode to the precision matrix, combine these two together to get a Gaussian distribution to approximate the true posterior. if we take

$$f_S = K\alpha_S$$

then optimizing f_S is equivalent to optimizing α_S. Suppose we get an optimal α_S, then the mode of f_S is $K\alpha_S$, the Hessian wrt f_S at $K\alpha_S$ is:

$$-\frac{1}{2}(K^{-1} + \text{diag}(y_i\sigma(-y_i f(x_i)))_{n\times 1}) := -\frac{1}{2}(W + K^{-1})$$

So picking such $K\alpha_S$ as mean, and

$$\Sigma_S = (W + K^{-1})^{-1}$$

as covariance of posterior process costrained on training set, the predictive distribution of a new data point x_* is as (3):

$$\mu(x_*) = k(x_*)^T \alpha_S$$
$$\sigma(x_*)^2 = k(x_*, x_*) - k(x_*)^T (K + W^{-1})^{-1} k(x_*)$$

Then the relative entropy can be evaluated in closed form. If we denote

$$A = I + W^{1/2} K W^{1/2}$$

Then [4]

$$D(Q||P) = \frac{1}{2}(\ln|A| + \text{Tr}(A^{-1}) + \alpha_S^T K_I \alpha_S - n)$$

3 Assumed Density Filtering and Sparse Approximation

Assumed Density Filtering keep the functional form of posterior process to be Gaussian. The approach only processes one example and update the posterior mean and covariance once per round. One view of it is that the classification problem is transformed into a regression problem, by transforming classification dataset into a equivalent regression dataset. Throughout this section, we use probit noise model instead of logistic noise model to maintain compuational tractability:

$$p(y_i|f(x_i)) = \Phi(\lambda y_i (f(x_i) + b))$$

In ADF, we want to approximate

$$\frac{p(f_S) \times p(y_1|f_1) \times \ldots \times p(y_n|f_n)}{Z}$$

sequentially using Gaussian distribution in each step.

The 0^{th} step is approximation of $p(f_S)$ using a Gaussian process $q_0(f_S) = N(\mu_0, \Sigma_0)$, minimizing $D(p(f_S)||q_0(f_S))$, which is trivial because $p(f_S)$ is already Gaussian.

The 1^{st} step is approximation of

$$p_1(f_S) = q_0(f_S) \times \frac{p(y_1|f_1)}{Z_1}$$

using a Gaussian process

$$q_1(f_S) = N(\mu_1, \Sigma_1)$$

minimizing $D(p_1(f_S)||q_1(f_S))$, which can be calculated using sufficient statistics matching because of property of exponential family, which will be given in detail in the next subsection.

Applying the same procedure on data point $(x_i, y_i), i = 2, 3, \ldots, n$, the n^{th} step is approximation of

$$p_n(f_S) = q_{n-1}(f_S) \times \frac{p(y_n|f_n)}{Z_n}$$

using a Gaussian process

$$q_n(f_S) = N(\mu_n, \Sigma_n)$$

minimizing $D(p_n(f_S)||q_n(f_S))$ in the same manner.

For computational efficiency, sometimes only a subset of training data points need to be used to update the posterior distribution. The key idea of [3, 1] is to keep track of the posterior process depending on only a subset of examples. There are two views of the updated process in [3](implicit updating posterior mean and covariance matrix on full training data), [1](build posterior mean and covariance function incrementally). They turn out to be equivalent, except differences in data point selection and deletion(See appendix A for details).

3.1 Update in IVM: implicit updating mean and covariance matrix of the full data

The idea of Informative Vector Machine(abbrev. IVM)[3] is to keep the posterior mean vector and covariance matrix of full data, although implicitly for computatonal efficiency. In sparse approximation, instead of processing all n data points, we process a selected subset $I \subset \{1, 2, \ldots, n\}$ of data points. The point selection rules are deferred in following sections. In i^{th} step of approximation, without loss of generality, we are going to add data point x_i into the model(if not, we can swap the selected n_i^{th} example with the original i^{th} example). Denote the $(i, i)^{th}$ element of Σ_j as $\zeta_{j,i}$. Because we are going to approximate the posterior with a Gaussian in each step, the sufficient statistics matching is essentially the first and second moments matching. These moments with respect to the posterior distribution can be evaluated in closed form, if we denote

$$Z_i(\mu_{i-1}, \Sigma_{i-1}) = \int N(f; \mu_{i-1}, \Sigma_{i-1}) p(y_i|f_i) df = \int N(f_i; \mu_{i-1,i}, \zeta_{i-1,i}) p(y_i|f_i) df_i \tag{4}$$

as the normalization factor in each round, then it turns out that the moment can be connected with the derivative of $\ln Z_i$ wrt μ_{i-1} (more specifically, only connect with first and second order derivative wrt their i^{th} coordinate, $\mu_{i-1,i}$), so that they can be evaluated in closed form [3]. Let

$$g_i = \frac{\partial \ln(Z_i)}{\partial \mu_{i-1,i}}, \nu_i = \frac{\partial^2 \ln(Z_i)}{\partial \mu_{i-1,i}^2}$$

So if we assume probit noise model, then denote:

$$u_{i-1,i} = c_{i-1,i}(\mu_{i-1,i} + b)$$

$$c_{i-1,i} = \frac{y_i}{\sqrt{\lambda^{-2} + \zeta_{i-1,i}}}$$

they are:

$$g_i = \frac{c_{i-1,i} N(u_{i-1,i}; 0, 1)}{\Phi(u_{i-1,i})}$$

$$\nu_i = g_i(g_i + u_{i-1,i} c_{i-1,i})$$

Then the update of μ and Σ only involves:

$$\begin{aligned} \mu_i &= \mu_{i-1} + g_i \Sigma_{i-1} e_i \\ \Sigma_i &= \Sigma_{i-1} + \nu_i \Sigma_{i-1} e_i e_i^T \Sigma_{i-1} \end{aligned} \tag{5}$$

where e_i is the i^{th} canonical basis vector.

Another view of this approach is we reduce the classification problem into a regression problem, by transforming the original classification example to a hypothetical regression example in round i to be $N(m_i; f_i, \beta_i^{-1})$

$$m_i = \frac{g_i}{\nu_i} + \mu_{i-1,i}$$

$$\beta_i = \frac{\nu_i}{1 - \nu_i \zeta_{i-1,i}}$$

Given the points selected (denoted as set $I = \{n_1, n_2, \ldots, n_{|I|}\}$), the posterior process only depends on the m_i, β_i values of the points $\{x_i\}$ selected. So we can safely ignore other data points in the training set and only concentrate on the set I, note that the complexity scales with $O(n|I|^2)$, instead of $O(n^2)$ if we process all the data points unselectively. It can be proven that

$$q_i(f_I) \propto N(f_I; 0, K_I) N(m_I; f_I, B_I^{-1})$$

The final approximation is

$$P(f_I|S) = N((K_I^{-1} + B_I)^{-1} B_I m_I, (K_I^{-1} + B_I)^{-1})$$

We denote all notations in resemblance with section 2:

$$\beta = (B_I^{-1} + K_I)^{-1} m_I$$

$$B = I + B_I^{1/2} K_I B_I^{1/2}$$

Then the relative entropy can be evaluated in closed form:

$$D(Q||P) = \frac{1}{2}(\ln |B| + \mathrm{Tr}(B^{-1}) + \beta^T K_I \beta - |I|)$$

So the predictive distribution of a testing point is:

$$\mu(x_*) = k_I(x_*)^T \beta$$

$$\sigma(x_*)^2 = k(x_*, x_*) - k_I(x_*)^T (K_I + B_I^{-1})^{-1} k_I(x_*)$$

3.2 Update in SOGP: incremental way to build the parameter of covariance

The idea of Sparse Online Gaussian Process (abbrev. SOGP)[1] is to keep track with the mean and covariance function of posterior process, which is equivalent to moment matching. Assume we have processed $x_1, \ldots, x_i$, $\mu_i(x)$ and $k_i(x, x')$ are the mean and covariance function of approximate posterior process $q_i(.)$, $k_{x,i} = (k(x, x_1), \ldots, k(x, x_i))^T$.

$$\mu_i(x) = k_{x,i}^T \alpha_i = k_x^T \begin{pmatrix} \alpha_i \\ 0 \end{pmatrix}$$

$$k_i(x, x') = k(x, x') + k_{x,i}^T C_i k_{x',i} = k(x, x') + k_x^T \begin{pmatrix} C_i & 0 \\ 0 & 0 \end{pmatrix} k_x$$

So using the reasoning of moment matching lemma, the first and second order moment of posterior should be [1]:

$$\begin{aligned} \mu_i(x) &= \mu_{i-1}(x) + g_i k_{i-1}(x, x_i) \\ k_i(x, x') &= k_{i-1}(x, x') + \nu_i k_{i-1}(x, x_i) k_{i-1}(x_i, x') \end{aligned} \tag{6}$$

g_i, ν_i are the first and second order derivatives of (4) wrt $\mu_{i-1,i}$, same as that of IVM. Then all extending updates can be presented in matrix form:

$$\begin{aligned} \alpha_i &= \begin{pmatrix} \alpha_{i-1} + g_i C_{i-1} k_i \\ g_i \end{pmatrix} \\ C_i &= \begin{pmatrix} C_{i-1} + \nu_i (C_{i-1} k_i)(C_{i-1} k_i)^T & \nu_i (C_{i-1} k_i) \\ \nu_i (C_{i-1} k_i)^T & \nu_i \end{pmatrix} \end{aligned} \tag{7}$$

3.3 The difference in the two approaches

3.3.1 The selection rule of IVM

The selection of training examples into active set is based on using differential entropy score, i.e. selecting the data point into the subset which make the posterior differential entropy smallest. Since we update the posterior based on one example each time, then the update can be evaluated efficiently: The point selection rule is to achieve minimum differential entropy in the updated distribution, to minimize $\ln|\Sigma_i|$, i.e. to maximize

$$-\ln|\Sigma_{i-1}^{-1}\Sigma_i| = -\frac{1}{2}\ln(1-\nu_{i,n_i}\zeta_{i-1,n_i})$$

3.3.2 The selection and deletion rule of SOGP

A major difference between SOGP and IVM is that SOGP is online, i.e. it has to decide whether to keep an example immediately it sees the example. In contrast, in each round, IVM can look at all training data in a batch to decide which example to add in the next round. If SOGP exceeds its memory constraint, it will delete an example in the active set. Since SOGP's online manner, one heuristic may be to do several sweeps of data to take full advantage of data. Note in this approach the posterior may highly overfit if #sweeps are large, because the example swept are not fresh.

If the new data point x_i lies approximately in the span of the selected examples(in feature space view):

$$k(x,x_i) = k_{x,i-1}^T\hat{e}_i$$

Then the updates(call it a reduced update) can be performed without extending the α and C matrix:

$$\alpha_i = \alpha_{i-1} + g_i C_{i-1}k_i + \hat{e}_i$$

$$C_i = C_{i-1} + \nu_i(C_{i-1}k_i + \hat{e}_i)(C_{i-1}k_i + \hat{e}_i)^T$$

If x_i does not exactly lies in span, the approximate $\hat{e}_i$ vector can be easily evaluated:

$$\hat{e}_i = K_{i-1}^{-1}k_i$$

If the number of point stored exceeds the memory constraint d, we need to perform a deletion, i.e. to delete a basis vector. Firstly we select the point that has the minimum square error if it is deleted, which turns out to be

$$i = \text{argmin}\,\frac{|\alpha_i|}{Q_{i,i}}$$

where $Q = K^{-1}$. Wlog, suppose $i = d+1$, it needs to delete the $(d+1)^{th}$ element because of memory constraint (otherwise swap the to-be-deleted element with the $(d+1)^{th}$ one), while the current parameters are as follows:

$$Q = \begin{pmatrix} Q^{(d)} & Q^* \\ Q^{*T} & q^* \end{pmatrix} = K^{-1}, C = \begin{pmatrix} C^{(d)} & C^* \\ C^{*T} & c^* \end{pmatrix} \alpha = \begin{pmatrix} \alpha^{(d)} \\ \alpha^* \end{pmatrix} \tag{8}$$

Then we perform the deletion, but still doing the reduced update on this data point to take advantage of it [1]:

$$\alpha = \alpha^{(d)} - \alpha^*\frac{Q^*}{q^*}$$

$$C = C^{(d)} + c^*\frac{Q^*Q^{*T}}{q^{*2}} - \frac{1}{q^*}[Q^*C^{*T} + C^*Q^{*T}]$$

$$Q = Q^{(d)} - \frac{Q^*Q^{*T}}{q^*}$$

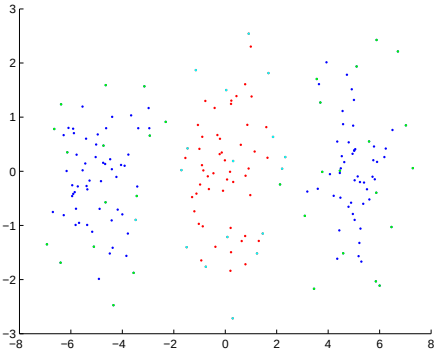


Figure 1: point selected by IVM

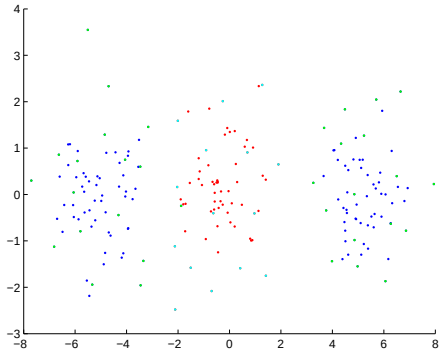


Figure 2: point selected by SOGP

3.4 Illustrative example in point selection

Here we present an illustrative synthetic example in a synthetic 2d data to demonstrate the point selection rule in the IVM as a sanity check of the experiments in next section. The problem is a binary classification one, with three components: normal distribution $N((-5,0),I_2)$, $N((0,0),I_2)$, $N((5,0),I_2)$, each component with mixing weights 1/3. We assign the component $N((0,0),I_2)$ to one class, and the rest to the other. The experiment uses RBF kernel, Setting the hyperparameters with bandwidth parameter 1, and function scale parameter 1, probit noise variance 1 and offset 0. Using 200 training data. Sparsity parameter $|I|$(#points selected in final approximation) is set 50. The red and blue points are from two classes, the green(resp. cyan) points are from blue(resp. red) class and marked as points selected in final approximation. The points selected by IVM(resp. SOGP) is shown in Figure 1(resp. Figure 2).

As shown by the figures, the "informative vectors" selected tend to lie close to the boundary of a certain class. What is more interesting, there are not only points close to decision boundary, but also points that are close to the boundary of support set.

4 Experiments Implementation

4.1 Testing the PAC-Bayesian Bounds of Gibbs classifier

In this project we conducted some experiments on UCI dataset: breast, bupa, haberman, hepatitis, hypo, pima. All experiments are based on nonoverlapping sets of 100 training examples and 200 testing examples, each dimension of features normalized to have mean 0 and standard deviation 1. All experiments uses RBF kernels, where the x-scale and y-scale factor is set to 1. Using the IVM [3], whose posterior relative entropy proved empirically favorable compared with full logistic regression based on full training set in [4]. The full logistic regression can also produce similar results, but much more time consuming.

According to PAC-Bayes Theorem (1), the estimate of upper bound of generalization Gibbs error depend on the empirical Gibbs error and the gap error bound. In the experiments, we set the IVM containing 30 active examples for a relatively sparse model. δ value is set to be 0.5. The following plots' x axis is testing error, and y axis is the estimated upper bound $D^{-1}_{Ber}(E_{w\sim Q}er_S(w), \frac{D(Q||P)+\ln\frac{n+1}{\delta}}{n})(D^{-1}(p,\epsilon) = \inf\{q \geq p : D_{Ber}(p||q) \geq \epsilon\})$, and the line is $y = x$ as a reference. The experiments are over 100 runs and each pair of (test error, estimated upper bound) is plotted as a point in the scatterplot. From the plots we see although some bounds tend to be trivial (exceeding 0.5), but they still have some linear relationship with the test error, although different method's strength of linear relationship are different.

Figures 3-8 provide a comparison among the three algorithms. For full logistic regression and SOGP, their Gibbs error is almost the same, both less than the IVM's Gibbs error. This is not surprising because they both incorporate full training data information into the posterior process, while IVM completely ignore the part of point not selected in the model. Taking a closer look at the upper bound estimate, we can see that the relative entropy term of IVM is slightly less than that of SOGP and full logistic

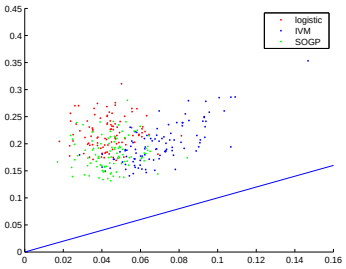


Figure 3: breast

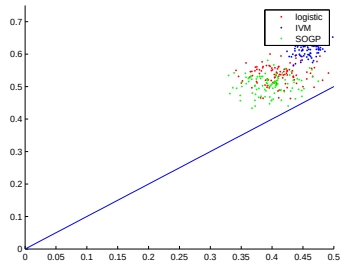


Figure 4: bupa

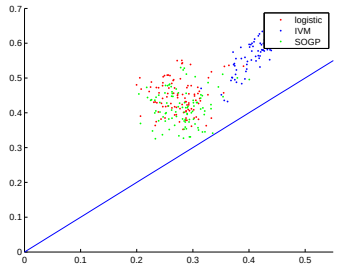


Figure 5: bupa

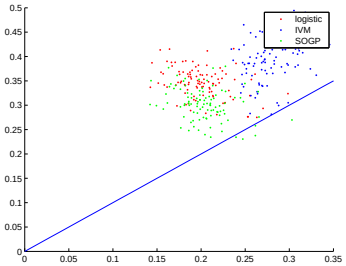


Figure 6: hepatitis

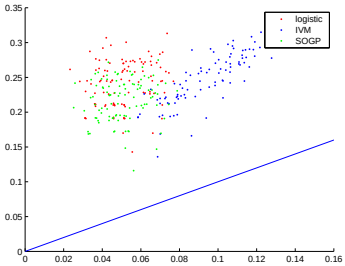


Figure 7: hypo

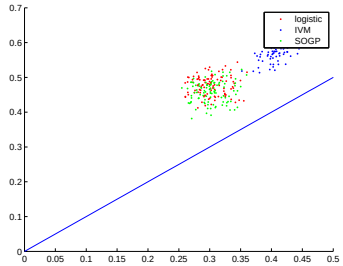


Figure 8: pima

regression, and the Gibbs training error is greater(not shown in report). In the datasets above, the empirical Gibbs error dominates. But in some medium scale and easier datasets(like MNIST 2v3), the gap error bound term dominates, we will see IVM is more favorable than full logistic regression because of dominance of relative entropy term.

4.2 Using validation set to determine hyperparameters

The following experiments are based on using IVM setting $|I| = 300$, with nonoverlapping sets of 3000 training examples and 3000 validation examples, using MNIST 2v3 dataset. The preprocessing might be different from [4] because we employ PCA and normalized every vector in preprocessing. We used a grid of parameter candidate set w, C for kernel

$$k(x, x') = C \exp\left(-\frac{w||x - x'||^2}{2d}\right)$$

d is the dimensionality of x. The range of w, C is $\{10^{-4}, 10^{-3}, \ldots, 10^{3}\} \times \{10^{1}, 10^{2}, \ldots, 10^{8}\}$. We list the Gibbs training and test error, 0-1 training and test error, gap error bound and Gibbs test upper bound estimate as follows (Tables 1-6):

Table 1: Gibbs training error

$w \backslash C$	10^1	10^2	10^3	10^4	10^5	10^6	10^7	10^8
10^{-4}	0.4994	0.4935	0.4424	0.2719	0.1269	0.0836	0.0712	0.0675
10^{-3}	0.4932	0.4412	0.2742	0.1281	0.0793	0.0724	0.0654	0.0495
10^{-2}	0.4395	0.2723	0.1284	0.0777	0.0689	0.0535	0.0282	0.0204
10^{-1}	0.2729	0.1220	0.0782	0.0510	0.0253	0.0167	0.0157	0.0160
10^{0}	0.1246	0.0615	0.0290	0.0169	0.0147	0.0143	0.0137	0.0140
10^{1}	0.0489	0.0245	0.0204	0.0196	0.0204	0.0197	0.0203	0.0201
10^{2}	0.2829	0.2752	0.2743	0.2742	0.2746	0.2768	0.2731	0.2726
10^{3}	0.4541	0.4487	0.4492	0.4486	0.4489	0.4487	0.4498	0.4492

Table 2: Gibbs test error

$w\backslash C$	10^1	10^2	10^3	10^4	10^5	10^6	10^7	10^8
10^{-4}	0.4998	0.4932	0.4411	0.2669	0.1204	0.0779	0.0665	0.0622
10^{-3}	0.4930	0.4398	0.2701	0.1214	0.0730	0.0671	0.0615	0.0488
10^{-2}	0.4377	0.2682	0.1220	0.0721	0.0654	0.0530	0.0354	0.0312
10^{-1}	0.2685	0.1152	0.0734	0.0512	0.0327	0.0284	0.0274	0.0271
10^{0}	0.1186	0.0612	0.0360	0.0290	0.0290	0.0284	0.0281	0.0276
10^{1}	0.0569	0.0400	0.0380	0.0369	0.0381	0.0373	0.0378	0.0378
10^{2}	0.3189	0.3147	0.3140	0.3139	0.3136	0.3155	0.3139	0.3118
10^{3}	0.4999	0.4999	0.4999	0.4999	0.5000	0.4999	0.5000	0.4999

Table 3: Gap error bound

$w\backslash C$	10^1	10^2	10^3	10^4	10^5	10^6	10^7	10^8
10^{-4}	0.0040	0.0044	0.0049	0.0068	0.0106	0.0155	0.0223	0.0298
10^{-3}	0.0040	0.0046	0.0063	0.0102	0.0152	0.0221	0.0304	0.0421
10^{-2}	0.0042	0.0060	0.0098	0.0150	0.0227	0.0347	0.0503	0.0600
10^{-1}	0.0056	0.0097	0.0156	0.0275	0.0442	0.0553	0.0578	0.0571
10^{0}	0.0100	0.0201	0.0368	0.0494	0.0532	0.0532	0.0541	0.0529
10^{1}	0.0280	0.0452	0.0498	0.0512	0.0507	0.0511	0.0507	0.0507
10^{2}	0.0458	0.0526	0.0534	0.0534	0.0535	0.0536	0.0534	0.0535
10^{3}	0.0461	0.0527	0.0534	0.0535	0.0535	0.0535	0.0535	0.0535

Table 4: Estimated Gibbs error upper bound

$w\backslash C$	10^1	10^2	10^3	10^4	10^5	10^6	10^7	10^8
10^{-4}	0.5453	0.5419	0.4934	0.3268	0.1824	0.1427	0.1402	0.1476
10^{-3}	0.5397	0.4901	0.3276	0.1818	0.1369	0.1412	0.1458	0.1395
10^{-2}	0.4864	0.3244	0.1813	0.1344	0.1371	0.1358	0.1164	0.1131
10^{-1}	0.3228	0.1736	0.1358	0.1215	0.1033	0.1002	0.1012	0.1015
10^{0}	0.1777	0.1229	0.1010	0.0947	0.0945	0.0932	0.0926	0.0919
10^{1}	0.1185	0.1036	0.1017	0.1019	0.1026	0.1020	0.1025	0.1024
10^{2}	0.4307	0.4331	0.4339	0.4338	0.4341	0.4365	0.4321	0.4318
10^{3}	0.6057	0.6109	0.6124	0.6119	0.6121	0.6120	0.6128	0.6123

Table 5: 0-1 training error

$w\backslash C$	10^1	10^2	10^3	10^4	10^5	10^6	10^7	10^8
10^{-4}	0.5013	0.4357	0.0507	0.0460	0.0453	0.0477	0.0443	0.0457
10^{-3}	0.3803	0.0587	0.0500	0.0480	0.0467	0.0490	0.0420	0.0250
10^{-2}	0.1030	0.0487	0.0487	0.0433	0.0423	0.0193	0.0103	0.0083
10^{-1}	0.0423	0.0497	0.0400	0.0180	0.0090	0.0053	0.0057	0.0060
10^{0}	0.0380	0.0167	0.0087	0.0053	0.0040	0.0027	0.0030	0.0043
10^{1}	0.0057	0.0017	0.0010	0.0007	0.0010	0.0010	0.0007	0.0003
10^{2}	0	0	0	0	0	0	0	0
10^{3}	0	0	0	0	0.0007	0	0	0

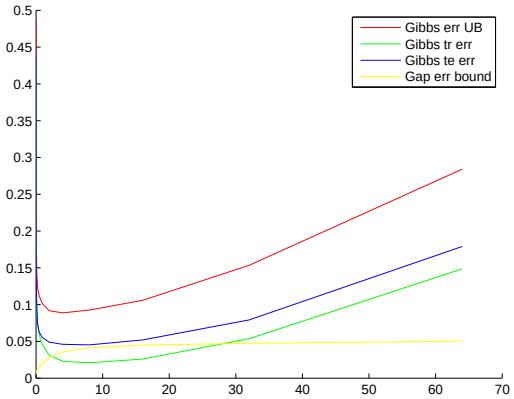
Table 6: 0-1 test error

$w\backslash C$	10^1	10^2	10^3	10^4	10^5	10^6	10^7	10^8
10^{-4}	0.5113	0.4353	0.0450	0.0397	0.0403	0.0383	0.0403	0.0390
10^{-3}	0.3583	0.0513	0.0423	0.0413	0.0383	0.0410	0.0373	0.0247
10^{-2}	0.1017	0.0443	0.0427	0.0403	0.0410	0.0207	0.0160	0.0147
10^{-1}	0.0407	0.0407	0.0353	0.0217	0.0150	0.0140	0.0123	0.0110
10^{0}	0.0330	0.0200	0.0113	0.0110	0.0113	0.0117	0.0120	0.0113
10^{1}	0.0113	0.0103	0.0103	0.0103	0.0097	0.0103	0.0097	0.0093
10^{2}	0.0093	0.0070	0.0077	0.0097	0.0097	0.0080	0.0090	0.0080
10^{3}	0.0370	0.0410	0.0427	0.0447	0.0460	0.0380	0.0403	0.0450

The tables shows the relationship of training and test error with respect to kernel paramameter. Seen from Gibbs test error, $w \approx 0.1, C \approx 10^8$ achieve the best Gibbs test error, which might exceed the limit of our grid search. It is interesting that when w is large, the Gibbs error is large, but the 0-1 error is still relatively small. What is more, the 0-1 training error monotonically decrease to 0 and 0-1 test error first decrease then increase, which gives a hint of overfitting as C fixed, and keep w increasing. On the other hand, both training Gibbs error and testing Gibbs error first decrease then increase, indicating the increasing uncorrelatedness of training data point and test data point, as w increasing.

Taking a closer look at each value, fix $C = 100$, let w varies, then we get a plot of relevant quantities as Figure 4.2. The figure shows the estimated Gibbs error upper bound is a good indicator of the true Gibbs error. It may be promising to use the upper bound to do hyperparameter selection as well, which may be a future direction.

Figure 9: red: PAC-Bayes upper bound, blue: test Gibbs error, green: training Gibbs error, yellow: gap error bound values



4.3 Using type II maximum likelihood to determine hyperparameters

There are 4 hyperparameters in the model: the horizontal/vertical scale of kernel, C and w, and the probit noise offset and variance b and λ. the type II maximum likelihood is a reasonable approach to determine them, because marginal likelihood automatically provide the trade-off of model fitting and model complexity. It turns out that using some approximation, all marginal likelihood (and their derivatives wrt hyperparameters) can be evaluated in closed form, thus grid search of hyperparameters(curse of dimensionality) can be avoided. But it is not clear whether the result here is comparable with that of previous section, because Gibbs error in that may not be a good criterion for model comparison.

4.3.1 Determining the kernel parameters

Unfortunately directly evaluating the full data likelihood is intractable, because it can be seen as an integration on a rectangle area over a Gaussian density. In ADF framework, since we have converted the classification examples to equivlent hypothetical regression example, maximizing the likelihood of these example is a reasonable choice, which can be evaluated in closed form:

$$p(m) = \int N(f; 0, K)N(m; f, B^{-1})df = N(m; 0, K + B^{-1})$$

Taking derivatives of $\ln p(m)$ wrt $\Sigma = K + B^{-1}$, we get

$$\frac{\partial \ln p(m)}{\partial \Sigma} = \frac{1}{2}(-\Sigma^{-1} + \Sigma^{-1} m m^T \Sigma^{-1})$$

then because $\Sigma = K + B^{-1}$ is only a linear relationship,

$$\frac{\partial \Sigma_{i,j}}{\partial C} = \exp\,(-\frac{w||x_i - x_j||^2}{2d}), \frac{\partial \Sigma_{i,j}}{\partial w} = C(-\frac{||x_i - x_j||^2}{2d}) \exp\,(-\frac{w||x_i - x_j||^2}{2d})$$

so the derivative can be evaluated:

$$\frac{\partial \ln p(m)}{\partial C} = \frac{\partial \ln p(m)}{\partial \Sigma} \cdot \frac{\partial \Sigma}{\partial C}, \frac{\partial \ln p(m)}{\partial w} = \frac{\partial \ln p(m)}{\partial \Sigma} \cdot \frac{\partial \Sigma}{\partial w}$$

We can alternate between optimizing the kernel parameter and the IVM to do hyperparameter selection.

4.3.2 Determining the noise parameters

maximizing $\ln p(m)$ seem hard, because the parameters only indirectly depend on b and λ value. One way is to provide a variational lower bound of the true likelihood $\ln p(y)$ and maximize the lower bound.

$$\begin{aligned}
&\ln p(y) \\
&= \ln (\int p(y_1|f_1) \dots p(y_N|f_N) p(f) df) \\
&= \ln (\int p(y_1|f_1) \dots p(y_N|f_N) p(f) \frac{q(f_1) \dots q(f_n)}{q(f_1) \dots q(f_n)} df) \\
&\geq \int \ln (\frac{p(y_1|f_1) \dots p(y_N|f_N) p(f)}{q(f_1) \dots q(f_n)}) q(f_1) \dots q(f_n) df \\
&= \sum_{n=1}^{N} \int \ln (p(y_n|f_n)) q(f_n) df_n + \text{const}
\end{aligned}$$

The relevant term can be upper bounded:

$$\sum_{n=1}^{N} \int \ln p(y_n|f_n) q(f_n) df_n \leq \sum_{n=1}^{N} \ln (\int p(y_n|f_n) q(f_n) df_n) = \sum_{n=1}^{N} \ln Z_n$$

Taking derivatives of $\sum_{n=1}^{N} \ln Z_n$ wrt u (recall notation in section 3), we get

$$\frac{\partial \ln Z_n}{\partial u_n} = \frac{N(u_n)}{\Phi(u_n)}$$

then because

$$\frac{\partial u_n}{\partial b} = c_n, \frac{\partial u_n}{\partial \lambda} = -\frac{1}{2} u_n c_n^2 (-2\lambda^{-3})$$

so the derivative can be evaluated:

$$\frac{\partial \sum_{n=1}^{N} \ln Z_n}{\partial b} = \sum_{n=1}^{N} g_n, \frac{\partial \sum_{n=1}^{N} \ln Z_n}{\partial \lambda} = \lambda^{-3} \sum_{n=1}^{N} g_n u_n c_n$$

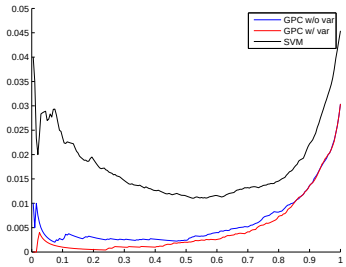


Figure 10: breast

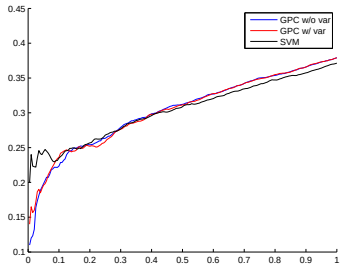


Figure 11: bupa

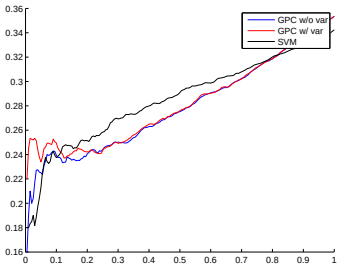


Figure 12: haberman

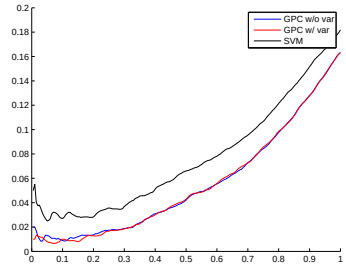


Figure 13: hepatitis

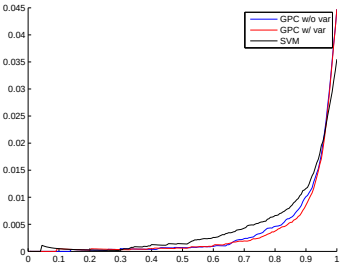


Figure 14: hypo

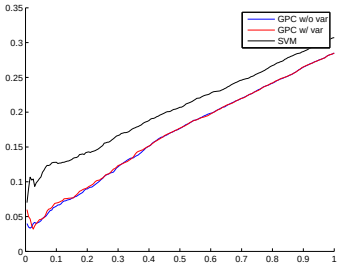


Figure 15: pima

The experiments are based on using IVM setting $|I| = 300$, with 3000 training examples, using MNIST 2v3 dataset, same as previous section. When considering only the two kernel hyperparameters (fixed $b = 0, \lambda = 1$), after 20 rounds of iteration the results have been stable, those selected are approximately: $w = 21.84$, C grows unbounded. When considering all four hyperparameters, it seems the b, λ parameters quite depend on the C values and does not converge.

4.4 Trials in using mean and standard deviation of posterior function value for selective classification

For selective classification problems, given a coverage c, we need to select c-fraction of examples to make decision so that its testing error is as small as possible. For a specific testing point (x_*, y_*), Gaussian Process produces not only the mean $\mu(x_*)$ but also the standard deviation $\sigma(x_*)$ of posterior function value of the $f(x_*)$ (which is clearly a Gaussian distribution), so intuitively if the distribution of $f(x_*)$ is not only have large absolute value, but also highly concentrated, it is good. The Gibbs error of such a point is

$$e_{Gibbs}(x_*, t_*) = \Phi(\frac{-y_*\mu(x_*)}{\sigma(x_*)})$$

For the ratio $|\frac{\mu(x_*)}{\sigma(x_*)}|$, if it is closer to zero, then we should regard it as more uncertain. So a heuristic may be sorting the testing examples in descending order of the ratio and always pick top c-quantile of the examples to be classified and reject the remaining examples to achieve coverage c.

we employ this strategy and compared it to using purely $|\mu(x_*)|$ values for sorting, and the traditional SVM margin approach. Below is the risk-coverage tradeoff curve, averaged over 100 runs of experiment, the setting are exactly the same as section 4.1: x axis is the coverage requirement Pr(makes a prediction), y axis is the risk, i.e. the estimate of Pr(the prediction mistakes|makes a prediction) in testing data. Note the comparison of GP with SVM may be unfair, because not only the sorting rule but also the classifier is different. The conclusions we draw from these figures are too weak; there is only some improvements in bupa dataset, and the other datasets are either unfair or indistinguishable.

References

[1] Lehel Csató and Manfred Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[2] Neil D. Lawrence, John C. Platt, and Michael I. Jordan. Extensions of the informative vector machine. In Joab Winkler, Mahesan Niranjan, and Neil D. Lawrence, editors, *Deterministic and Statistical Methods in Machine Learning*, volume 3635 of *Lecture Notes in Computer Science*, pages 56–87. Springer, 2004.

[3] Neil D. Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse gaussian process methods: The informative vector machine. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 609–616. MIT Press, 2002.

[4] Matthias Seeger. Pac-bayesian generalisation error bounds for gaussian process classification. *Journal of Machine Learning Research*, 3:233–269, 2002.

A The equivalence of the two approaches

Since IVM and SOGP are both rooted from the ADF framework, processing one data at a time, it is no surprise that they are equivalent. At first glance the update formula of the two approaches are quite different. IVM maintains the covariance matrix of full training data(although implicitly), while SOGP maintains the parameters (related to only a subset of data) of mean and covariance function. For completeness, we point out the equivalence of these two views.

The IVM maintains the posterior mean and covariance of the training example implicitly: in each step it updates as (5) while in Sparse Online maintains the C_i and α_i to track posterior mean function and covariance function explicitly, in each step is updates as (6), more specifically the parameters are updated as (7).

We can prove by induction that $\mu_i = (\mu_i(x_s), s = 1, 2, \ldots, n)$, $\Sigma_i = (k_i(x_s, x_t), s, t = 1, 2, \ldots, n)$.

A.1 From mean/covariance function to mean/covariance matrix of training data

Base case: for $i = 0$, $\mu(x) \equiv 0, k_0(x, x') \equiv k(x, x')$, so plugging in the training data X, $\mu_0 = 0, \Sigma_0 = K$.
Inductive case: assume the function view is equivalent to matrix view before $i-1$, we have $k_{i-1}(X, x_i) = \Sigma_{i-1}e_i$. Then if we have the function update, the matrix update can be performed by plugging the x, x' values to be the whole training data,

$$\mu_i = \mu_{i-1} + g_i k_{i-1}(X, x_i) = \mu_{i-1} + g_i \Sigma_{i-1} e_i$$

$$\Sigma_i = \Sigma_{i-1} + \nu_i k_{i-1}(X, x_i) k_{i-1}(x_i, X) = \Sigma_{i-1}(\nu_i e_i e_i^T)\Sigma_{i-1}$$

A.2 From mean/covariance matrix of training data to mean/covariance function

Base case: for $i = 0$, $\mu_0 = 0, \Sigma_0 = K$ with no data seen, so the posterior mean and covariance function stay the same: $\mu(x) \equiv 0, k_0(x, x') \equiv k(x, x')$.
Inductive case: assume the function view is equivalent to matrix view before $i-1$, we have $k_{i-1}(X, x_i) = \Sigma_{i-1}e_i$. Then if we have the matrix update, the α_t, C_t parameters of the function update can be figured out. Because

$$f_x | f_S \sim N(f_x; k_x^T K^{-1} f_S; k(x, x) - k_x^T K^{-1} k_x)$$

So the mean function and covariance function deducted by the matrix update is:

$$\mu_i(x) = k_x^T K^{-1} \mu_i = k_x^T k^{-1} \mu_{i-1} + k_x^T K^{-1} \Sigma_{i-1} g_i e_i$$

$$k_i(x) = k(x, x) - k_x^T K^{-1} k_x + k_x^T K^{-1} \Sigma_i K^{-1} k_x = k(x, x) - k_x^T K^{-1} k_x + k_x^T K^{-1} \Sigma_{i-1} K^{-1} k_x + k_x^T K^{-1} \Sigma_{i-1} \nu \Sigma_{i-1} K^{-1} k_x$$

Because $k_{i-1}(x, x_i) = k_x^T K^{-1} \Sigma_{i-1}$, the predictive process is equivalent to directly function updates.

A.3 The equivalence of the update formulae

We can also show that the update formulae are equivalent. The update essentially keep the connection of

$$\mu_i = K \begin{pmatrix} \alpha_i \\ 0 \end{pmatrix}$$

$$\Sigma_i = K + K \begin{pmatrix} C_i & 0 \\ 0 & 0 \end{pmatrix} K$$

In this view it can be easily proved by induction that the two kinds of updates:
(1) update wrt μ, Σ:

$$\mu_i = \mu_{i-1} + g_i \Sigma_{i-1} e_i$$

$$\Sigma_i = \Sigma_{i-1} + \nu_i \Sigma_{i-1} e_i e_i^T \Sigma_{i-1}$$

(2) update wrt α, C(note that α_i, C_i has one more dimension than α_{i-1}, C_{i-1}):

$$\begin{pmatrix} \alpha_i \\ 0 \end{pmatrix} = \begin{pmatrix} \alpha_{i-1} \\ 0 \\ 0 \end{pmatrix} + g_i \begin{pmatrix} C_{i-1} k_i \\ 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} C_i & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} C_{i-1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \nu_i \begin{pmatrix} C_{i-1} k_i \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} (C_{i-1} k_i)^T & 1 & 0 \end{pmatrix}$$

are equivalent.