A Refined MCMC Sampling from RKHS for PAC-Bayes Bound Calculation

Li Tang[1,2], Zheng Zhao[1], Xiu-Jun Gong[1,3,*]
[1]School of Computer Science and Technology, Tianjin University, Tianjin 300072, China
[2]Information science and technology Department, Tianjin University of Finance and Economics, Tianjin 300222, China
[3]Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin 300072, China
Email: tangli0831@yeah.net, zhengzh@tju.edu.cn, gongxj@tju.edu.cn

***Abstract*—PAC-Bayes risk bound integrating theories of Bayesian paradigm and structure risk minimization for stochastic classifiers has been considered as a framework for deriving some of the tightest generalization bounds. A major issue in practical use of this bound is estimations of unknown prior and posterior distributions of the concept space. In this paper, by formulating the concept space as Reproducing Kernel Hilbert Space (RKHS) using the kernel method, we proposed a refined Markov Chain Monte Carlo (MCMC) sampling algorithm by incorporating feedback information of the simulated model over training examples for simulating posterior distributions of the concept space. Furthermore, we used a kernel density method to estimate their probability distributions in calculating the Kullback-Leibler divergence of the posterior and prior distributions. The experimental results on two artificial data sets show that the simulation is reasonable and effective in practice.**

***Index Terms*—PAC-Bayes Bound, Reproducing Kernel Hilbert Space (RKHS), Markov Chain Monte Carlo (MCMC), Support Vector Machine (SVM)**

I. Introduction

Computational Learning Theory (CLT) provides a formal framework for the analysis of the performance of learning algorithms [1]. Probably Approximately Correct (PAC) Learning Theory proposed by Valiant in 1984, derived from CLT, advanced the framework by associating learning problem with computational complexity [2]. PAC bound is characterized by its VC-dimension of the hypothesis space with the assumption of independent identically distribution (i.i.d.). PAC-Bayesian bounds, first proposed by McAllester [3] and improved by Seeger [4], Langford [5] and Shawe-Taylor [6], introduce a prior partition of the hypothesis space and apply non-uniform treatment of the hypotheses.

PAC-Bayes risk bound integrating theories of Bayesian paradigm and structure risk minimization for stochastic classifiers has been considered as a framework for deriving some of the tightest generalization bounds. A lot of research proved that PAC-Bayes bound is a tighter bound for the classifiers and a better way for analyzing the generalization performance of learning algorithms [4-7]. Various well established learning algorithms can be justified and even improved in the PAC-Bayes framework [8-10]. PAC-Bayes bounds were previously applied to classification [4-10], but over the last few years the theory has been extended to cluster analysis [11], Gaussian Process regression [12], density estimation [13], and problems with non iid data [14-15].

Nevertheless, some problems still remain to be solved. First of all, the theorem of PAC-Bayes bound provides probably approximately correct mathematical form for the upper bound of a generalization error rate. But the most of items in the inequality of theorem cannot be directly obtained and estimated from the experimental results. Consequently, the analytic expression of the bound is very difficult to deduce. It leads to the narrow application of PAC-Bayes bound. Meanwhile, the calculation of PAC-Bayes bound requires that prior and posterior distributions of concepts output by the classifiers must be conformed to multivariate Gaussian distributions with unit covariance matrix, this leads that it is available only for linear SVM and Gaussian process.

In our previous study, we formulated the concept space as Reproducing Kernel Hilbert Space (RKHS) using the kernel method [16]. We further demonstrated that the RKHS can be constructed using the linear combination of kernels, and the support vectors and their corresponding weights of SVM describe the complexity of concept space. Therefore, the calculation of PAC-Bayes bound can be simulated by sampling weights of support vectors in RKHS.

In this paper, we proposed a refined Markov Chain Monte Carlo (MCMC) sampling algorithm for simulating posterior distributions of the concept space by incorporating feedback information of the simulated model over training examples. Furthermore, we used a kernel density method to estimate their probability distributions in calculating the Kullback-Leibler divergence of the posterior and prior distributions. The experimental results on two artificial data sets showed that the simulation is reasonable and effective in practice.

Manuscript received June 1, 2013; revised September 4, 2013; accepted September 15, 2013.

This research was partly supported by the Natural Science Funding of China (Grant No. 61170177), National Basic Research Program of China (Grant No. 2013CB32930X), innovation funding of Tianjin University and the funding of scientific research development project of Tianjin University of Finance and Economics (No. Q1114).

* Corresponding author: Xiu-Jun Gong

© 2014 ACADEMY PUBLISHER
doi:10.4304/jcp.9.4.930-937

The paper is organized as the followings: section II introduces PAC-Bayes bound and its application on linear classifier and concept space formulating; section III proposes the refined MCMC sampling algorithm and kernel density estimation; section IV shows the realization of algorithm and experimental results; section V draws the conclusion.

II. PRELIMINARIES

A. PAC-Bayes Bound and Its Application on Linear Classifier

We recall the PAC-Bayes bound for the binary classification problems presented in [3-5]. Given the input space X consists of an arbitrary subset of R^n and the output space $Y = \{-1, +1\}$. Consider that a pair (x, y) with $x \in X$ and $y \in Y$ as an example, which is drawn from a fixed distribution on $X \times Y$. In the PAC-Bayes setting, a concept h is defined by a distribution $q(h)$ over the hypothesis space. For given m examples, we are interested in the gap between the expected generalization error $Q_D = E_{(x,y)\in D} I(h(x) \neq y)$ and the expected empirical error $\hat{Q}_S = \frac{1}{m} \sum_{(x_i,y_i)\in S} (I(h(x_i) \neq y_i))$, where $I(\alpha)=1$ if predicate α is true and 0 otherwise. Q_D denotes the probability that the classifier h misclassifies an instance x chosen from the distribution D. Meanwhile, the empirical error $\hat{Q}_S$ means the probability that the classifier h misclassifies an instance x chosen from the sample S. The gap between them could be parameterized by the Kullback-Leibler divergence.

$$KL(p \,\|\, q) = q \ln \frac{q}{p} + (1-q) \ln \frac{1-q}{1-p} \tag{1}$$

Theorem 1. (PAC-Bayes bound) For any distribution D, arbitrary prior P and confidence $\delta \in (0,1]$, then all posteriors Q satisfy inequality (2) with probability at least $1-\delta$ over samples $S \sim D^m$,

$$KL(\hat{Q}_s \,\|\, Q_D) \leq \frac{KL(Q \,\|\, P) + \ln(\frac{m+1}{\delta})}{m} \tag{2}$$

where KL is the Kullback-Leibler divergence.

$$KL(Q \,\|\, P) = E_{h\sim Q} \ln(Q(h) / P(h)) \tag{3}$$

For a given learning algorithm and the training data set, $\hat{Q}_S$ and the right expression is fixed in the inequality (2). Consequently, the inequality (2) provides the upper bound of average true error rate Q_D, which plays an important role in the assessment of generalization performance.

The PAC-Bayes bound was applied to the SVM [5-6]. Assumed that P and Q follow the independent identical Gaussian distributions with $P = N(\vec{p}, \mu_1, \Sigma_1)$ and $Q = N(\vec{q}, \mu_2, \Sigma_2)$. Then the KL value can be computed using equation (4).

$$KL(Q \,\|\, P) = \frac{1}{2}(\mu_1 - \mu_2)^T \Sigma_2^{-1} (\mu_1 - \mu_2) + \frac{1}{2} \log \frac{|\Sigma_1|}{|\Sigma_2|} + \frac{1}{2} tr\{\Sigma_1 \Sigma_2^{-1} - I_d\} \tag{4}$$

Furthermore, we assume that both Σ_1 and Σ_2 are unit matrices, and the KL value can be replaced by $\Delta\mu^2/2$.

Theorem 2. (PAC-Bayes bound for linear classifiers [5]) For all distributions D, for all classifiers given by w and $\mu>0$, for any $\delta \in (0,1]$, then the inequality (5) holds with probability $1-\delta$ over the samples of the training data.

$$KL(\hat{Q}_s(w,\mu) \,\|\, Q_D(w,\mu)) \leq \frac{\frac{\Delta\mu^2}{2} + \ln(\frac{m+1}{\delta})}{m} \tag{5}$$

There are two main difficulties in calculating the average true error rate Q_D.

Firstly, Most of learning algorithms output a concept, other than the distribution of the concept. As a result, the posterior distribution Q is difficult to estimate. Furthermore, the prior distribution P need be predefined. Therefore, the KL divergence between two distributions is not easy to calculate in the inequality (2). A traditional way is to make some assumptions like the inequality (5). The KL values replaced by $\Delta\mu^2/2$ are under the assumption that the prior and posterior distribution of the concept should be multivariate Gaussian distribution with unit variance matrix. However, the assumption is too strict to realization.

Secondly, the estimation of $\hat{Q}_S$ is not easy. The error rate of examples in concept space is a continuous segmented function because of the limited number of training examples; moreover, the posterior distribution of concept space is unknown.

B. Concept Space Formulating

In our previous research, we have formulated the concept space as a RKHS using the kernel method and demonstrated that the support vectors and their corresponding weights of SVM describe the complexity of concept space [16]. We further proved that the distributions of the concepts in RKHS can be simulated using the distributions of the weight vectors from the outputs of the SVM algorithm. Therefore, the calculation of PAC-Bayes bound can be simulated by sampling weights of support vectors in RKHS [16]. Let us just simply recall our previous results.

Theorem 3. (Representer theorem [17]) Let RH to be a RKHS with the reproducing kernel function $K : X \times X \to R$. For arbitrary function $L : R^n \to R$ and a strictly monotonically increasing real-valued function $\Omega : R^n \to R$, if the expression(6) is well defined, then there exist $\alpha_1, \ldots \alpha_k \in R$ that make (7) minimize (6).

$$J^* = \min_{f\in RH} J(f) = \min_{f\in RH} \{\Omega(\|f\|^2_{RH}) + L(f(x_1), f(x_2), \ldots, f(x_n))\} \tag{6}$$

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i K(\cdot, x_i) \tag{7}$$

© 2014 ACADEMY PUBLISHER

Theorem 4. (Riesz Representation theorem) Let H be a Hilbert space, and let H^* denote its dual space, consisting of all continuous linear functions from H into the field R. If x is an element of H, then the function φ_x defined by

$$\varphi_x(y) =< y, x > \quad \forall y \in H \tag{8}$$

where $<\cdot,\cdot>$ denotes the inner product of the Hilbert space, is an element of H^*. The Riesz representation theorem states that every element of H^* can be written uniquely in this form.

Assume that data is linear separable after nonlinear projection, linear hyper plane can be expressed as linear function F which satisfies

$$F(\alpha_1 x_1 + \alpha_2 x_2) = \alpha_1 F(x_1) + \alpha_2 F(x_2) \tag{9}$$

According to Riesz Representation theorem, there's $f_F \in H$ for arbitrary linear function F in Hilbert space to satisfy

$$F(f) =< f, f_F >, \forall f \in H \tag{10}$$

Linear function can be expressed by the inner product of vectors. Like the finite Euclidean space, the hyper plane is expressed as

$$F(f) =< f, g > +b \tag{11}$$

where g is a weight vector, and b is a intercept.

Similarly with linear SVM, the target function is:

$$\min \frac{1}{2}\|g\|^2$$

$$\text{subject to} \quad y_i(< f_i, g > +b) \geq 1, i = 1,...n. \tag{12}$$

According to Representer theorem, the optimal solution is expressed as

$$g^* = \sum_{i=1}^{n} \alpha_i f_i \tag{13}$$

The problem of finding the optimal solution g^* in the infinite Hilbert space is transformed into finding the optimal solution $\alpha^* = (\alpha_1^*, \cdots \alpha_n^*)^T$ in n dimensional Euclidean space.

According to the above proof, finding a minimal function in RKHS is equal to finding optimal values $\alpha^* = (\alpha_1^*, \cdots \alpha_n^*)^T$ in Euclidean space. A concept to be learned is a function in the RKHS. Consequently, learning a concept is identical to finding optimal values $\alpha_1^*, \alpha_2^* \cdots \alpha_n^* (\alpha_i^* \in R)$, and sampling the function in RKHS is equivalent to sampling the weight vectors $\alpha^* = (\alpha_1^*, \cdots \alpha_n^*)^T$.

Theorem 5. A concept to be learned for SVM algorithm can be described by the weights of support vectors.

From the theorem 5, we can know that different support vectors and their weight vectors correspond to different concepts that learning algorithms aim to optimize. The number of support vectors reflects the complexity of the concept space.

III. PAC-Bayes Bound Calculation

A. Refined MCMC Sampling Algorithm

The major issue for calculating the KL values is the unknown prior (P) and posterior (Q) distribution of the concept space. In our previous research, we used two types of sampling methods (random sampling and MCMC sampling) to simulate the posterior distribution of the concept space. Both of them did not take into the consideration feedback information of the simulated model over training examples. Intuitively, sampling in RKHS should be done around weight vectors with high accuracy models over training examples. Consequently, the accuracy of the models can be considered as their feedback information.

In this paper, we refine MCMC algorithm to incorporate the accuracy of the simulated model over training examples as the feedback information into the MCMC sampling process. The feedback of previous sampling can play a guiding role in the next sampling. Since the refined MCMC sampling might destroy the assumption that traditional MCMC generated sequences are in accord with a normal distribution, we use kernel density estimation method to calculate posterior probability density of the concept space.

MCMC methods are a class of algorithms for sampling from probability distributions based on constructing a Markov chain [18]. The general idea of Metropolis-Hastings algorithm (MHA) is to generate a series of samples that are linked in a Markov chain. Supposed that MHA can draw samples from a probability distribution $p(a)$ and the most recent sample value is a_j, MHA generates a new proposal value a' with the proposal distribution $v(a'|a_j)$, which is the conditional probability of generating a proposal sample a' given the most recent sample a_j. We calculate the value r using (14).

$$r = \min\{\frac{p(a')}{p(a_j)} \frac{v(a_j \mid a')}{v(a' \mid a_j)}, 1\} \tag{14}$$

In MHA, whether the proposal value a' is accepted depends on the value of r. We accept the proposal value with the probability r and refuse it with the probability 1-r. If refused, that means the next value a_{j+1} is equal to a_j, else, $a_{j+1} = a'$.

The proposal distribution is usually supposed to be a symmetric distribution, leading to $\frac{v(a_j \mid a')}{v(a' \mid a_j)} = 1$. From the equation (14), the value of r relies on $\frac{p(a')}{p(a_j)}$. The ratio of probability density is defined by (15).

$$R_1(a', a_j) = \frac{p(a')}{p(a_j)} \tag{15}$$

© 2014 ACADEMY PUBLISHER

According to (15), whether to accept the proposal value depends on the ratio of the probability density. In the MHA, we need not calculate the probability density of samples explicitly, but the ratio of the probability density. Because of unknown probability distribution of samples, we cannot calculate their probability density and the ratio of probability density directly.

Because sampling in RKHS should be done around weight vectors with high accuracy models over training examples, we propose to incorporate the accuracy of the simulated model over training examples as feedback information into the MCMC process to approximate the ratio of probability density.

According to the proposal value *a'* and the previous sample a_j, we combine them with the support vectors to establish their respective models (m(*a'*) and m(a_j)). With the simulated models, we execute the classification and prediction by SVM algorithm, and calculate the accuracy of models respectively: Acc(m(*a'*)) and Acc(m(a_j)). The ratio of their accuracy is defined.

$$R_2(a', a_j) = \frac{Acc(m(a'))}{Acc(m(a_j))} \tag{16}$$

As above formulation, the distributions of the concepts in RKHS can be simulated using the distributions of the weight vectors from the outputs of the SVM algorithm. Obviously, the concept with high accuracy for SVM classification has the high probability in the sampling of RKHS, we can deduce that sampling in RKHS should be done around weight vectors with more accurate models. Therefore, probability density of samples is approximately proportional to the accuracy of their models. As a result, the ratio of probability density can be simulated by the ratio of the accuracy of their models, and we make use of the ratio of the accuracy of models $R_2(a', a_j)$ to approximate the ratio of probability density $R_1(a', a_j)$. After the approximation of $R_1(a', a_j)$ by $R_2(a', a_j)$, we calculate the value of *r*, which can decide to accept or refuse the proposal value. In this way, the accuracy of models, as the feedback information, is incorporated into the MCMC sampling process, to determine the value of *r* and guide the next sampling.

With the implementation of refined MCMC sampling method, we sample and obtain a sequence of samples. The pseudo code for the refined MCMC sampling algorithm is listed in algorithm (1).

Algorithm 1: samplingRefinedMCMC
Input: Alpha, SupportVector, $T=\{(x_i, y_i)|x_i \in R^n, y_i \in \{0,1\}, i=1,...,m\}$;
/*weight vectors, support vectors and training set*/
Output: TQExamples;
/*getting the posterior samples of concepts */
1 mu, var = initializingMeanVariance(Alpha);
/* initializing the mean and variance */
2 TM = buildingMCMC(mu, var); /* building probability models for the MCMC instance TM*/
3 TM.sample(1) = Alpha; /*initializing the samples*/
4 For i = 1 To 50^3 /* defining sampling times*/
5 ProposalValue = initializingProposal(TM.sample(i)); /*initializing the proposal value according to weight vectors*/
6 AlphaModel = buildingModel(Alpha, SupportVector); /*building the model according to weight vectors and support vectors*/
7 ProposalModel = buildingModel(ProposalValue, SupportVector); /*building the proposal model according to proposal value and support vectors*/
8 AAlpha = SVMPredict(T, AlphaModel);/*calculating the accuracy of previous model by SVM prediction*/;
9 AProposal = SVMPredict(T, ProposalModel); /*calculating the accuracy of proposal model by prediction of SVM*/;
10 RValue = AProposal/AAlpha; /*returns the ratio of the accuracy of proposal and previous models*/
11 TM.sample(i+1)= MHA(TM.sample(i), ProposalValue, RValue); /*getting new weight vector with MHA*/
12 TQExamples = TM.sample(i+1);
13 End For
14 return TQExamples;

B. Kernel Density Estimation

After the implementation of above steps, we get a sequence of samples. Then we estimate the probability density of these samples in order to calculate the Kullback-Leibler divergence. However, the distribution of samples is unknown, and its probability density function is also unknown. In statistics, the non-parametric method is always suitable for the solution of this case.

There are two kinds of method in the non-parametric way, including Nearest Neighbor Estimation (NNE) and Kernel Density Estimation (KDE). The general idea of NNE is to calculate the distances between the samples. Nevertheless, the calculation of distances is complex, and the selection of coefficient is difficult. Obviously, the NNE is not fit for this case. We adopt KDE for the probability density function of samples.

Suppose that *A* denotes a continuous random variable and we have random samples a_1, a_2,...a_n drawn from a probability density $f_A(a)$, and we wish to estimate f_A at a point a_0 [19-21]. For simplicity, we assume that $A \in R$. KDE with width λ is used

$$\hat{f}_A(a_0) = \frac{1}{n\lambda}\sum_{i=1}^{n} K_\lambda(a_0, a_i) \tag{17}$$

We can deduce that the samples approximately conform to a normal distribution centered weight vectors in general. In this case, a popular choice for K_λ is the Gaussian kernel $K_\lambda(a_0,a) = \Phi(|a-a_0|/\lambda)$. Letting Φ_λ denotes the Gaussian density with mean zero and standard-deviation λ, we has the form

$$\hat{f}_A(a_0) = \frac{1}{n}\sum_{i=1}^{n} \Phi_\lambda(a - a_i) \tag{18}$$

The generalization of the Gaussian KDE amounts to the expression (19).

© 2014 ACADEMY PUBLISHER

$$\hat{f}_A(a_0) = \frac{1}{n(2\lambda^2\pi)^{\frac{1}{2}}} \sum_{i=1}^{n} e^{-\frac{1}{2}(\|a_i - a_0\|/\lambda)^2} \quad (19)$$

In this way, the probability density of posterior distribution has been obtained by the Gaussian KDE.

C. Calculation of KL Values and PAC-Bayes Bound

We assume that the prior distribution of concepts follows the normal distributions with mean value 0 and unit covariance. The probability density function of prior distribution can be acquired easily.

Since the probability density of prior and posterior distribution has been known, KL values can be calculated. According to the equation (3), KL value is defined by

$$KL(Q \mid P) = sum(\ln(Q \mid P)) / len(Q) \quad (20)$$

The average true error Q_D can be calculated using a binary search approach in our previous research [16]. The pseudo code for the calculation of KL values and average true error is listed in algorithm (2).

Algorithm 2: imPacBayesBound
Input: T = $\{(x_i, y_i)|x_i \in R^n, y_i \in \{0,1\}, i=1,...,m\}$ /*training set*/
Output: KL; /*Kullback-Leibler divergence of the posterior and prior distribution of concepts */
Output: QD; /* true average error*/
1 Alpha, SupportVector = trainingSVM(T);
/*getting weight vectors and support vectors by training dataset T */
2 TPExamples = samplingNormal(Alpha); /*getting prior examples of concepts using normal distributions*/
3 TQExamples = samplingRefinedMCMC(Alpha, SupportVector, T);
/*getting the examples by refined MCMC sampling*/
4 TPDensity = normalDensity(TPExamples);
/*getting the prior probability density*/
5 TQDensity = GaussianKDE(TQExamples);
/*using Gaussian kernel density estimate for posterior */
6 KL = calcKL(TQDensity, TPDensity); /*calculating the KL value of the posterior and prior distribution */
7 QS = calcEAverageError(L,T); /*calculating the true average error of examples on T using learning L */
8 QD = binarySearch(KL, QS, m, delta) ;
/*calculating the true average error */

IV. Experiment and Results

A. Data Sets

Two artificial datasets are used to test the capability of algorithm (2). The numbers of support vectors in two data sets are 3 and 4 respectively. See Table I and Table II for more details.

TABLE I.
DATA SET 1 WITH 3 SUPPORT VECTORS

Labels	Support Vector	Weight
+1	0.1 0.2 0.7 0.5 0.8	0.3602982226576067
+1	0.9 0.4 0.2 0.1 0.5	0.6397017773423932
-1	0.6 0.3 0.5 0.3 0.3	-1

TABLE II.
DATA SET 2 WITH 4 SUPPORT VECTORS

Labels	Support Vector	Weight
+1	0.1 0.2 0.7 0.5 0.8	0.3032936982010008
+1	0.9 0.4 0.2 0.1 0.5	0.5639045380403556
+1	0.1 0.6 0.8 0 0.4	0.1328017637586437
-1	0.6 0.3 0.5 0.4 0.3	-1

B. Results

Sampling the posterior distribution for 50^3 times by refined MCMC method, we can get the following Fig. I.

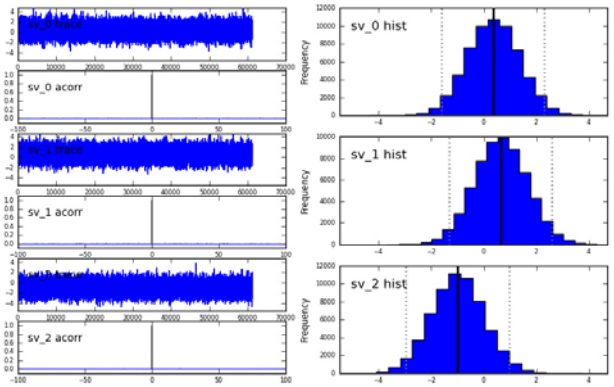

Figure I. Refined MCMC Sampling for 3 support vectors

In the Fig. I, the left part is the trace and autocorrelation graphs in which the horizontal axis denotes the time of sampling, and vertical axis denotes corresponding values and autocorrelations. The right part is the histogram in which the horizontal axis denotes sampled values, and vertical axis denotes the number of corresponding value in sampling.

From the Fig. I, we see that the samples approximately conform to a normal distribution centered on the weight vectors in general. Therefore, we select Gaussian kernel density estimation to estimate the probability density function of the posterior distribution.

In the same way, we draw the MCMC sampling figure for the data set 2 as Fig. II.

© 2014 ACADEMY PUBLISHER

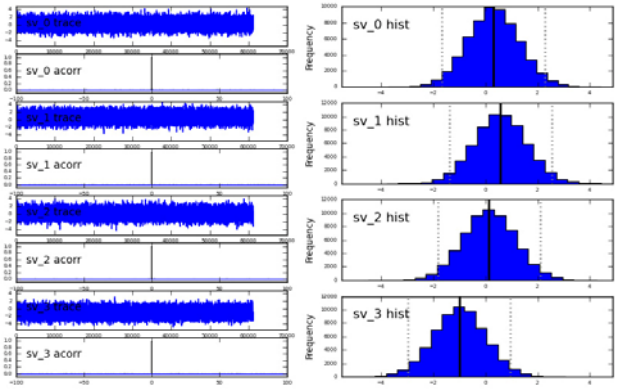

Figure II. Refined MCMC Sampling for 4 support vectors

From the Fig. I and Fig. II, the samples approximately conform to a normal distribution centered on the weight vectors in general. Gaussian kernel density estimation is used to estimate the probability density function of the posterior distribution.

For each dataset, we calculate the KL values and PAC-Bayes bound by using five methods. Method A is traditional method, in which KL value is replaced by $\Delta\mu^2/2$. Method B is the random sampling. Method C is MCMC sampling on the assumption that both the posterior distribution and the prior distribution of concept are Gaussian distributions. We use the Gaussian density function to derive the probability density of prior and posterior distribution. Method D is a refined MCMC sampling on the assumption that both the posterior distribution and the prior distribution of concept are Gaussian distributions. We derive the probability density of prior and posterior distribution by the use of Gaussian density function. Method E is a refined MCMC sampling on the assumption that the prior distribution of concept is Gaussian distributions and posterior distribution is unknown. We adopt the kernel density estimation to derive the probability density of posterior distribution. The refine MCMC algorithm incorporates feedback information of the simulated model over training examples into the MCMC sampling process. The results are list in Table III and Table IV.

TABLE III.
PAC- BAYES BOUNDS WITH 3 SUPPORT VECTORS

method	KL	Q_D
A	0.85805	0.98409524405
B	0.770088307332 ± 1.14142E-05	0.444191090041 ± 1.52612E-05
C	0.767719757170 ± 1.1465E-05	0.445411034882 ± 9.66074E-07
D	0.770637950491 ± 1.61005E-05	0.445443167785 ± 1.1494E-06
E	0.791498109954 ± 1.36854E-05	0.444191090041 ± 1.52612E-05

(The value after ± is the variance of corresponding value for 10 times samplings)

TABLE IV.
PAC- BAYES BOUNDS WITH 4 SUPPORT VECTORS

method	KL	Q_D
A	0.99405	0.932905573621128
B	0.712929570965 ± 6.395E-06	0.427533307167 ± 1.57426E-07
C	0.712208985045 ± 1.37862E-05	0.427239717021 ± 8.86314E-07
D	0.713856734954 ± 2.80551E-05	0.427616031464 ± 3.15914E-07
E	0.785934743757 ± 3.46177E-05	0.427533307167 ± 1.57426E-07

(The value after ± is the variance of corresponding value for 10 times samplings)

From the tables III and IV, we can see that KL values and Q_D values in four sampling methods are smaller than ones using the traditional method, in which KL value is replaced by $\Delta\mu^2/2$. It shows that we can optimize the KL divergence and derive the tighter PAC-Bayes bound by the sampling method.

The four sampling methods get the similar KL values and Q_D values. It implies that the sampling process is stable. PAC-Bayes bound calculated by the MCMC sampling method is an effective and tighter risk bound.

First, we think about the MCMC sampling method and refined MCMC sampling method. Both method C and method D are on the assumption that both the posterior distribution and the prior distribution of concept are Gaussian distributions. Method C adopts MCMC sampling method while method D uses the refined MCMC sampling method by incorporating the feedback information of the simulated model over training examples into the MCMC sampling process. From the tables III and IV, we get the similar KL values and the similar PAC-Bayes bound in method C and method D. It proves that the refined MCMC sampling method by incorporating the accuracy of models as the feedback information is reasonable and effective. The ratio of probability density can be approximately simulated by the ratio of the accuracy of their models.

Furthermore, let us focus on the Gaussian density function and Gaussian KDE. The posterior distribution and the prior distribution of concept are assumed to be Gaussian distributions in method C and method D. In this case, we use the Gaussian density function to obtain the density of posterior distribution. From the table III, their KL values approximately equal to 0.77. From the table IV, the KL values almost equal to 0.71. The posterior distribution is unknown in method E. Under the unknown posterior distribution, Gaussian KDE is used for estimation of density. From the tables III, the KL values approximate to 0.79 by use of method E. From the table IV, the KL values are roughly equivalent to 0.78. According to the results, KL values on the assumption of Gaussian distribution are smaller than the KL values with the unknown posterior distribution. It means KL values calculated by Gaussian density function are a little

© 2014 ACADEMY PUBLISHER

smaller than KL values derived by Gaussian KDE. Nevertheless, we can get the tighter PAC-Bayes bound by Gaussian KDE from the above results. Gaussian KDE is fit for the case of destroying the assumption of Gaussian posterior distribution. In this case, it is the closest to the actual posterior distribution, and it can describe the reality more precisely. This leads to wider application of PAC-Bayes bound.

The variances of KL values in refined MCMC sampling method are larger than the other two sampling method. It might imply that refined MCMC sampling has better generalization performance than the others. The experiments show that the simulation is reasonable and effective in practice.

Although sampling by refined MCMC method quickly converges to the posterior distribution, a large quantity of samples can reflect the overall posterior distribution. The total sampling points should be more than 50^3 in the experiments. Nevertheless, when the dimension of space increases, the size of sampling points is in a geometric growth.

V. Conclusion

In this paper, we formulate the concept space as a RKHS by using the kernel method and propose a refined MCMC sampling algorithm by incorporating feedback information of the simulated model over training examples for simulating posterior distributions of the concept space. Furthermore, we used a kernel density method to estimate probability density of posterior distributions in calculating the KL values. In the experiment, five methods are used for the calculation of KL values and PAC-Bayes bound. The results show that the simulation is reasonable and effective in practice, leading to wide application of PAC-Bayes bound.

Acknowledgment

This research was supported in part by the Natural Science Funding of China under grant number 61170177, National Basic Research Program of China under grant number 2013CB32930X, innovation funding of Tianjin University and the funding of scientific research development project of Tianjin University of Finance and Economics (No. Q1114).

References

[1] M. H. G. Anthony, N. L. Biggs, *Computational Learning Theory*, Cambridge: Cambridge University Press, 1997, pp. 76–98.
[2] L. G. Valiant, A theory of the learnable, Communications of the ACM, vol. 27, pp.1134–1142, 1984.
[3] D. A. McAllester, "Some PAC-Bayesian theorems", Machine Learning, vol. 37, no.3, pp.355–363, 1998.
[4] M. Seeger, "PAC-Bayesian generalisation error bounds for Gaussian process classification", Journal of Machine Learning Research, vol. 3, pp.233-269, 2002.
[5] J. Langford, "Tutorial on practical prediction theory for classification", Journal of Machine Learning Research, vol. 6, pp.273-306, 2005.
[6] A. Ambroladze, E. Parrado-Hernandez, J. Shawe-Taylor, "Tighter PAC-Bayes bounds", In Proceedings of Neural Information Processing Systems, pp.9–16, 2006.
[7] J. Shawe-Taylor, D. Hardoon, "PAC-Bayes analysis of Maximum Entropy Classification", In Proceeding of 12th International Conference on Artificial Intelligence and Statistics, pp.480–487, 2009.
[8] L. François, M. Mario, "PAC-Bayes Risk Bounds for Stochastic Averages and Majority Votes of Sample-Compressed Classifiers", Journal of Machine Learning Research, vol. 8, pp.1461-1487, 2007.
[9] A. Pierre, "PAC-Bayesian Bounds for Randomized Empirical Risk Minimizers", Mathematical Methods of Statistics, vol. 17, no. 4, pp.279-304, 2008.
[10] E. Morvant, S. K. Co, L. Ralaivola, "PAC-Bayesian Generalization Bound on confusion Matrix for Multi-Class Classification", In 29th International Conference on Machine Learning, pp. 815–822, 2012.
[11] Y. Seldin, N. Tishby, "PAC-Bayesian Analysis of Co-clustering and Beyond", Journal of Machine Learning Research, vol. 11, pp.3595–3646, 2010.
[12] T SUZUKI, "PAC-Bayesian bound for Gaussian process regression and multiple kernel additive model", JMLR: Workshop and Conference Proceedings 23, 2012. http://jmlr.csail.mit.edu/proceedings/papers/v23/suzuki12/suzuki12.pdf
[13] M. Higgs, J. Shawe-Taylor, "A PAC-Bayes bound for tailored density estimation", In 21st International Conference on Algorithmic Learning Theory, vol. 6331, pp.148–162, 2010.
[14] L. Ralaivola, M. Szafranski, G. Stempfel, "Chromatic PAC-Bayes bounds for non-iid data", In Proceeding of 12th International Conference on Artificial Intelligence and Statistics, pp.416–423, 2009.
[15] G. Lever, F. Lavioette, "Distribution-Dependent PAC-Bayes Priors", In 21st International Conference on Algorithmic Learning Theory, pp.119–133, 2010.
[16] L. Tang, H. Yu, X. J. Gong, On the PAC-Bayes Bound Calculation based on Reproducing Kernel Hilbert Space, Applied Mathematics and Information Sciences, vol. 7, no. 2, pp.817–822, 2013.
[17] B. Schölkopf, R. Herbrich, A. J. Smola, "A generalized representer theorem", In Proceedings of the 14th Annual Conference on Computational Learning Theory, pp.416–426, 2001.
[18] Y. Gao, C. J. Zhang, L. Y. Zhang, "Comparison of GARCH Models based on Different Distributions", Journal of Computers, vol. 7, no. 8, pp.1967-1973,2012.
[19] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning: Data mining, inference, and prediction, Berlin: Springer, 2009, pp.35-57.
[20] N. Cristianini, J. Shawe-Taylor, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.
[21] Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

Li Tang was born in Hebei province, on August 31th, 1979. She is a PhD candidate in the School of Computer Science and Technology of Tianjin University. She received her master degree majored in Computer Application from Tianjin Polytechnic University in March, 2006. Her research interests

© 2014 ACADEMY PUBLISHER

include machine learning and data mining.

She has been working in Department of Information Science and Technology, Tianjin University of Finance and Economics since March, 2006. She is a lecturer and has published several research papers indexed by EI and SCI.

Zheng Zhao was born in 1948. He got his master degree from Stanford University in 1987. His research interests include machine learning and data mining.

He is a professor in the School of Computer Science and Technology of Tianjin University. He has published over 45 research papers on the international journals and conferences including IEEE transactions covering data mining.

Xiu-Jun Gong is an associate professor in the School of Computer Science and Technology of Tianjin University. He received his Ph. D degree majored in software and theory of computer science from the Institute of Computing, Chinese Academy of Science in 2002. His research interests include Bayesian learning theory, machine learning, distributed information system and bioinformatics.

He worked at the National University of Singapore and the Institute for Infocomm Research (I2R) as research and visiting fellow respectively from 2002 to 2003. He moved to the Nara Institute of Science and Technology in Japan as a research associate from May, 2003 to March, 2006. He came back China and joined the Tianjin University in May, 2006.

Dr. Gong has published over 35 research papers on the international journals and conferences including IEEE transactions covering data mining, bioinformatics and grid computing. He also serves as the members of several international conferences and reviewers of journals including Briefings in Bioinformatics, Computer Journal of Chinese, and tec.

© 2014 ACADEMY PUBLISHER