Applied Mathematics & Information Sciences
An International Journal

On the PAC-Bayes Bound Calculation based on Reproducing Kernel Hilbert Space

Li Tang[1,2],*Hua Yu*[1] *and Xiu-Jun GONG*[1]

[1] School of Computer Science and Technology, Tianjin University, Weijin Rd No. 92, Nankan, Tianjin, 300072, China
[2] School of Information Science and Technology, Tianjin University of Finance and Economics, Zhujiang Rd No. 25, Hexi, Tianjin, 300222, China

Received: 7 Jun. 2012; Revised 21 Sep. 2012; Accepted 23 Sep. 2012
Published online: 1 Mar. 2013

Abstract: PAC-Bayes risk bound combining Bayesian theory and structure risk minimization for stochastic classifiers has been considered as a framework for deriving some of the tightest generalization bounds. A major issue for calculating the bound is the unknown prior and posterior distributions of the concept space. In this paper, we formulated the concept space as Reproducing Kernel Hilbert Space (RKHS) using the kernel method. 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 outputs describe the complexity of concept space. Therefore the calculation of PAC-Bayes bound can be simulated by sampling weights of support vectors in RKHS. The experimental results using random and Markov Chain Monte Carlo (MCMC) samplings showed that the simulation is reasonable and effective in practice.

Keywords: PAC-Bayes bound, Reproducing Kernel Hilbert Space, Markov Chain Monte Carlo, Support Vector Machine.

1. Introduction

With the revival of artificial intelligence in 1980s, Computational Learning Theory (CLT) and Statistical Learning Theory (SLT) have gained huge progress [1,2]. Among those achievements, PAC (Probably Approximately Correct) theory, proposed by Leslie Valiant in 1984[3], has advanced the framework of machine learning into computational complexity. PAC bounds is characterized by its VC-dimension of the hypothesis space with the independent identically distribution (i.i.d.) assumption. However, unlike usual PAC bounds, PAC-Bayesian bounds, first proposed by McAlleste[4] and improved by Langford[5], apply non-uniform treatment of the hypotheses by introducing a prior partition of the hypothesis space.

PAC-Bayes bound generalizing the Occam?s razor bound for supervised algorithms which output a distribution over classifiers rather than just a single classifier has been considered as a framework for deriving some of the tightest generalization bounds. Many researches demonstrated that PAC-Bayes bound is a tighter bound for the classifier and a better way for the analysis of the generalization performance of learning algorithm[6–9]. Various well established learning algorithms can be justified in the PAC-Bayes framework and even improved. PAC-Bayes bounds were originally applicable to classification[10], but over the last few years the theory has been extended to regression[11], density estimation[12] , and problems with non iid data[13].

However, some challenges still remain. First, the PAC-Bayes bound theorem provides probably approximately correct mathematical form for the upper bound of a generalization error rate. But the most of items in the formula cannot be directly estimated from the experimental results, the analytic expression of the bound is very difficult to realize. This leads to the narrow applications of PAC-Bayes bound. Furthermore, the calculation of PAC-Bayes bound requires that prior and posterior distributions of concepts of the classifiers outputs must be conformed normal distributions with constant covariance matrix, this leads that it is applied only for linear classifier SVM and Gaussian process.

In this study, we first highlighted the difficulties in general PAC-Bayes bound calculations. Then, we formulated the concept space as Reproducing Kernel Hilbert Space

* Corresponding author: e-mail: gongxj@tju.edu.cn

(RKHS) using the kernel method. 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. Finally, we present an algorithm for calculating the PAC-Bayes bound.

The paper is organized as the followings: section 2 overviewed PAC-Bayes Bound and its practical difficulties in the specific calculations, section 3 presented the theoretical foundation, section 4 proposed the algorithm for the calculation of PAC-Bayes bound, section 5 presented the correspond experiments, and section 6 concluded the paper.

2. Overview of the PAC-Bayes bound

In this section, we recall the main PAC-Bayes bound for the binary classification problems presented in [4,5]. Considered that a binary classification problem where the input space X consists of an arbitrary subset of R^n and the output space $Y = \{-1,+1\}$. An example is an input-output pair (x,y) where $x \in X$ and $y \in Y$. Each example is drawn from a fixed, but unknown, distribution on $X \times Y$.

In the PAC-Bayes setting, a classifier h (also called a concept) is defined by a distribution $q(h)$ over the hypothesis space. Each classification is carried out according to a hypothesis sampled from $q(h)$. 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. The gap will be parameterized by the Kullback-Leibler divergence.

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

In fact, Q_D denotes the probability that the classifier h misclassifies an instance x chosen from the distribution D . Meanwhile, the empirical error means the probability that the classifier h misclassifies an instance x chosen from the sample S.

Theorem 1.*For an arbitrary D, arbitrary prior P and confidence* $\delta \in (0,1)$*,we have:*

$$\begin{aligned} &Prob_{S\sim D^m} \\ &\{\forall Q : KL(\hat{Q}_S \parallel Q_D) \leq \frac{KL(Q \parallel P) + ln(\frac{m+1}{\delta})}{m}\} \\ &\geq 1-\delta \end{aligned} \tag{2}$$

where KL is the Kullback-Leibler divergence and Q is the posterior distributions of the classifier h.

The theorem 1 indicates that: 1) Probably Correctness of (2) might be caused by the noise in training examples, for an example, the small number of training examples cannot cover the real distributions of the concepts; 2) Approximately Correctness might be caused by the system errors of learning algorithms, for instance, the huge concept space cannot be enumerated totally by learning algorithms. Both of the correctness can be manually predefined by the confidence degree δ.

In the following sections, we only focus on the inner part of (2).

$$KL(\hat{Q}_S \parallel Q_D) \leq \frac{KL(Q \parallel P) + ln(\frac{m+1}{\delta})}{m} \tag{3}$$

In the left of inequality (3), if $\hat{Q}_S$ is fixed, Q_D is monotonically increasing. Because $\hat{Q}_S$ and the right expression on the inequality are fixed for a given learning algorithm and a training data set, the inequality gives the upper bound of average true error rate Q_D .

The difficulties for calculating the average true error rate Q_D come from two aspects. Firstly, most of learning algorithms output a concept, other than the distribution of the concept and the posterior distribution Q is difficult to estimate. Meanwhile, the prior distribution P also need be predefined. Therefore, the calculation for KL values between two distributions is not easy. A traditional way is to make some assumptions. For an example, assumed that P and Q follow independent identical normal distributions with $P \sim N(\vec{p}, \mu_1, \Sigma_1)$ and $Q \sim N(\vec{q}, \mu_2, \Sigma_2)$. Then, the KL value can be computed using equation (4).

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

Further assumed that Σ_1 and Σ_2 both are unit matrices, the KL value can be replaced by $\frac{\Delta\mu^2}{2}$.

Secondly, the estimation of $\hat{Q}_S$ is not easy, because (1) the limited number of training examples, which leads that the example error rates in concept space is a continuous segmented function; 2) the posterior distribution of concept space is unknown.

3. Theoretical foundations

3.1. SVM Kernel function

SVM (Support Vector Machine) is a machine learning approach proposed by Vapnik based on structural risk minimization principle of statistics learning theory.

The PAC-Bayes bound can be particularized for the case of linear classifiers in the following way. The m training patterns define a linear classifier that can be represented by the following equation: $c(x) = sign(w^T \varphi(x))$, where $\varphi(x)$ is a nonlinear projection to a certain feature

space where a linear classification actually takes place, and w is a vector that determines the separating plane from the feature space.

For any vector w we can define a stochastic classifier in the following way: we choose the distribution $Q = Q(w,\mu)$ to be a spherical Gaussian with identity covariance matrix centered on the direction given by w at a distance μ from the origin. Moreover, we can choose the prior $P(c)$ to be a spherical Gaussian with identity covariance matrix centered on the origin. The performance of classifiers can be bounded by the form of expression (3). Here, $Q_D(w,\mu)$ is the true error of the stochastic classifier and $\hat{Q}_S(w,\mu) = E_m\tilde{F}(\mu\gamma(\bar{x},y))$ is the average over the m train examples, where $\gamma(\bar{x},y) = \frac{y\bar{x}\cdot\bar{w}}{\|\bar{w}\|\|\bar{x}\|}$ is the normalized margin of the train patterns and $\tilde{F}(x) = 1 - F(x)$ is the inverse of the cumulative normal distribution $F(x)$.

In SVM, the function $K(\cdot,\cdot)$ or the projection $\Phi(\cdot)$ is chosen to project the input space X into a finite or infinite dimensional Hilbert space.

Definition 1.*(Kernel function)X is a subset of R^n, the function $K(x,x')$ defined in $X \times X$ is called a kernel function, if there is a map $\Phi : X \to H$ from the input space to a Hilbert space such that (5).*

$$K(x,x') = (\Phi(x)\cdot\Phi(x')) \tag{5}$$

In the form of kernel, the output of SVM learning algorithm can be expressed by the following form:

$$c(\mathbf{x}) = sign(\sum_{i=1}^{m}\alpha_i K(\mathbf{x}_i,\mathbf{x})) \tag{6}$$

where K is the kernel function, x_i is called support vector if $\alpha_i \neq 0$, α_i is the weight of corresponding support vector.

Theorem 2.*(Mercer theorem)Let X be a compact subset of R^n , K is a continuous real-value symmetric function on $X \times X$, and $L_2(x)$ is a second order integral real-valued function space on X , the following two statements are equal:*

(1) The integral operator (7) is positive semi-defined.

$$\int_{X\times X} K(x,x')f(x)f(x')dxdx^i \geq 0 \ \ \forall f \in L_2(x) \tag{7}$$

(2) $K(\cdot,\cdot)$ can be expressed as a uniform convergent sequence of ψ on $X \times X$.

$$K(x,x') = \sum_{t=1}^{\infty}\lambda_t\psi_t(x)\psi_t(x') \tag{8}$$

Where λ_t and $\psi_t \in L_2(x)$ are the eigenvalue and its corresponding unit eigenfunction of the integral operator $T_k f = (T_k f)(\cdot) = \int_X K(\cdot,x')f(x')dx'$.

The mercer theorem shows that $K(x,x') = (\Phi(x),\Phi(x'))$ ($\Phi(x) = (\sqrt{\lambda_1}\psi(x_1),\sqrt{\lambda_2}\psi(x_2),\cdots)^T$) is a kernel function.

Definition 2.*(Gram matrix) for a given function $K : X \times X \to R$ and $(x_1,x_2,\cdots,x_n) \in X$? the matrix $G = \{(g_{ij} = K(x_i,x_j))\}$ is called the gram matrix about $x_1,x_2,\cdots,x_n$.*

It is easy to show that the necessary and sufficient condition that the symmetrical function $K(x,x')$ is a kernel function defined in $X \times X$ is that the gram matrix of the function is positive semidefinite, for any $(x_1,x_2,\cdots,x_n) \in X$.

3.2. Reproducing Kernel Hilbert Space

In section 3.1, we have known that the function Φ maps the input data to a Hilbert space, and also mercer theorem gives its formal expression. However, to get its specific form, we have to calculate the eigenvalues and their corresponding eigen-functions of the integral operator T_k , which is a trivial task in practice.

From mercer theorem, we can prove that the set defined in (9) is a vector space.

$$RH = \{(f(x))|f(x) = \sum_{i=1}^{k}\alpha_i K(x,x_i)\} \tag{9}$$

for all $(x_1,\cdots,x_n) \in X$ *and* $(\alpha_1,\cdots,\alpha_k) \in R$

And for the inner product, defined in (10), of two elements $f(\cdot) = \sum_{i=1}^{k_f}\alpha_i K(\cdot,x_i) \in RH$ and $(\cdot) = \sum_{j=1}^{k_g}\beta_j K(\cdot,y_j) \in RH$, the vector space is complete. So, the vector space RH is also a Hilbert space.

$$f * g = \sum_{i=1}^{k_f}\sum_{j=1}^{k_g}\alpha_i\beta_j K(x_i,y_j) \tag{10}$$

Now we can conclude that the kernel function maps Φ into a Hilbert space. RH is so-called the Reproducing Kernel Hilbert Space(RKHS). In this sense, K is called the reproducing kernel.

RKHS admits a well-defined structure that the target function of an optimization problem can be represented as a finite linear combination of kernel products. The Riesz Representation theorem gives its formal description.

Theorem 3.*(Riesz Representation theorem) Let RH to be a RKHS with the reproducing kernel function $K : X \times X \to R$. For any function $L : R^n \to R$ and any non-decreasing function $\Omega : R^n \to R$, if the expression (11) is well defined, then there exist $\alpha_1,\cdots,\alpha_n(\alpha_i \in R)$ that make (12) minimizing (11) .*

$$\begin{aligned} J^* &= \min_{f\in RH} J(f) \\ &= \min_{f\in RH}\{\Omega(\|f\|^2{}_{RH}) + L(f(x_1),f(x_2),...,f(x_n))\} \end{aligned} \tag{11}$$

$$f(\cdot) = \sum_{i=1}^{n}\alpha_i K(\cdot,x_i) \tag{12}$$

This theorem implies that finding a minimal function f^* in RKHS is equal to finding optimal values $\alpha_1, \alpha_2, \cdots, \alpha_n (\alpha_i \in R)$ in Euclidean space. In another words, the optimal values $\alpha_1, \alpha_2, \cdots, \alpha_n$ well describe the structure of the function f^* using kernel method.

Let's go back SVM learning algorithm again. We can easily know that the function Φ maps examples into Hilbert space, while the kernel function K maps examples into RKHS. A concept for learning algorithms to be learned is an element in RKHS. Therefore, learning a concept is identical to finding optimal values $\alpha_1, \alpha_2, \cdots, \alpha_n (\alpha_i \in R)$.

Recall the equation (6), we can easily draw the following conclusion.

Theorem 4.*A concept to be learned for SVM algorithm can be described by the weights of support vectors.*

From the theorem, 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.

Algorithm 1: calcPacBayesBound

Input: $T = \{(x_i, y_i) | x_i \in R^n, x_i \in \{0,1\}, i = 1 \ldots m\}$ //training set

Output: KL ; /* KL value of the prior and posterior distributions of concepts */

Output: QD ; /* true average error */

```
1 Alpha=trainingSVM (T) ;      /* getting weight vector of the support vectors by training data set T */
2 QS=calcEAverageError(L,T) ; /* calculating the true average error of examples on T using learning L.  */
3 TPExamples=samplingNormal() ;      /* get prior examples of concepts using normal distributions */
4 TQExamples=samplingAlpha(Alpha) ;    /* getting the true examples of concepts by sampling weight vector Alpha */
5 KL=calcKL(TQExamples, TPExamples) ;
  /* calculating KL value of the distributions of TQExamples and TPExamples */
6 QD=binarySearch(QS, KL, m, delta) ;
  /* calculating /true average error */
```

4. PAC-Bayes bound calculation

The major issue for calculating the true average error Q_D in (3) is the unknown prior (P) and posterior (Q) distributions of the concept space. A simple solution is to replace the KL value by $\frac{\Delta\mu^2}{2}$. Through above theoretical analysis, it cannot well describe the structure of concept space. Instead, the support vector and their weight vectors are well suited to the situation. Different support vectors and their weight vectors correspond to different concepts that learning algorithms aim to optimize. Therefore, the distributions of the concepts in RKHS can be simulated using the distributions of the weight vectors from the outputs of the SVM algorithm. In this section, we will present how to estimate the distributions of concepts in RKHs by ones of the weight vectors through sampling weight vectors in Euclid space for computing the true average error Q_D .

Supposed that there are m training examples in a dataset T, the main steps are illustrated in algorithm (1) to calculate true average error Q_D of a learning algorithm L.

The explanations of algorithm *calcPacBayesBound* are the followings:

Step (1) gets the weight vector of the support vectors by training data set T using SVM algorithm.

Step (2) calculates the average example error using learning algorithm L on dataset T.

Step (3) to step (5) are used to calculate the KL value of the prior and posterior distributions of concepts in RKHS. Supposed that there are k support vectors and their weights are considered as multiple random variables following multivariate normal distributions. The concepts in RKHS are simulated by these random variables. We sample the prior examples of concepts by the assumption that weight variables with mean value 0 and unit covariance follow multivariate normal distributions. And the posterior examples of concepts are sampled around the centers of weights with unit covariance.

We use two types of sampling methods: random sampling and Markov Chain Monte Carlo (MCMC) sampling.

Step (6) calculates the average true error using a binary search approach. Because is monotonically increasing if is fixed in the inequality (3), we intend to seek the largest value satisfying inequality (3) as the true average error. The pseudo code for the binary search method is listed in algorithm (2).

5. Experiment results

5.1. Data sets

We use two datasets for testing the capability of algorithm calcPacBayesBound in calculating PAC-Bayes bound. The numbers of support vectors in the two data sets are 3 and 4 respectively. See Table (1) and (2) for more details.

5.2. Results

To show that weight variables follow multivariate normal distributions after MCMC sampling, we draw trace graphs and histograms of 3 weight variables in data set (1) over

Algorithm 2: binarySearch

Input: QS, KL, m, delta
Output: QD

```
eps = 0.0001 ;
a=QS ;
b=(KL+ln((m+1)/delta))/(m+1) ;
lower = upper = a ;
while (kl(a, upper) ≤ b || upper ≤ 1- eps/2) do
    upper = upper + (1-upper)/2 ;
if (upper ≥ 1 − eps/2) then
    return 1 ;
while (upper − lower ≥ eps) do
    if (kl(a, lower + (upper − lower)/2) ≥ b)) then
        upper= lower+ (upper-lower)/2
    else
        lower=lower+ (upper-lower)/2
return upper ;
```

Table 1: data set with 3 support vectors

Label	Support vector	weight
1	0 0.2 0.7 0.5	0.4065091465768443
1	0.9 0.4 0.2 0.1	0.5934908534231558
-1	0.6 0.3 0.5 0.3	-1

Table 2: data set with 4 support vectors

Label	Support vector	weight
1	0 0.2 0.7 0.5	1
1	0.9 0.4 0.2 0.1	0.3633408445028599
1	0.1 0.6 0.8 0	0.1447171435460205
-1	0.6 0.3 0.5 0.3	0.4919420119511196

10^6 times sampling (Figure 1). From the figure, we notice that all the values of autocorrelation for the three variables are almost are 0. It implies that they follow a multivariate normal distribution. The positions of the largest values in histogram are initial values of weights. Therefore, examples for posterior concepts by MCMC sampling meet our requirements.

For each dataset, we calculate PAC-Bayes bounds using 10^5 and 10^6 examples using random and MCMC samplings. The KL and Q_D values for two datasets are given in Table (3) and (4) respectively

From the tables, we can see that Q_D values in both sampling methods are smaller than ones using the traditional method, in which KL value is replaced by $\frac{\Delta\mu^2}{2}$. Both of sampling methods get similar Q_D values. With more examples sampled the variances of KL values changed more slowly. This is agreed to our intuition that the concepts tend to be stable when enumerated enough examples. The variances of KL values in MCMC sampling are larger than ones in random sampling. This might imply that MCMC sampling has better generation performance than random sampling.

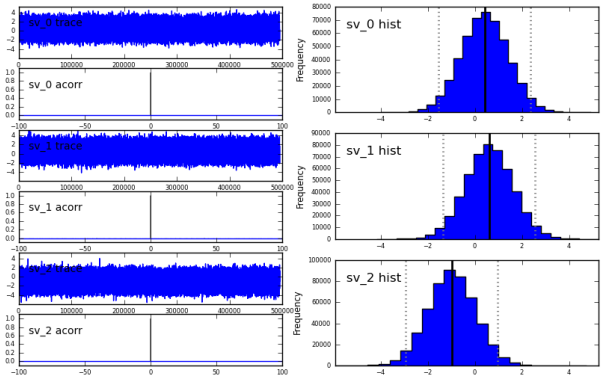

Figure 1: MCMC sampling graph (the left part is the trace and autocorrelation graphs in which the horizontal axis denotes the time of sampling, vertical axis denotes corresponding values and autocorrelations. The right part is the histogram in which the horizontal axis denotes sampled values, vertical axis denotes the number of corresponding value in sampling)

Table 3: PAC-Bayes bounds with 3 support vectors

	Mean and variance of KL value		Q_D	
	10^5	10^6	10^5	10^6
A	0.85805		0.984095244	
B	0.759694899 ±1.00592E-05	0.758760291 ±1.18045E-06	0.3414 71321	0.3361 00227
C	0.75920889 ±3.64688E-05	0.75973941 ±1.94675E-06	0.3414 71321	0.33610 0227

(A-KL replaced by $\frac{\Delta\mu^2}{2}$; B-Random sampling; C-MCMC sampling, the value after ± is the variance of corresponding value for 100 times samplings)

Table 4: PAC-Bayes bounds with 4 support vectors

	Mean and variance of KL value		Q_D	
	10^5	10^6	10^5	10^6
A	0.00005		0.97287178	
B	0.695865957 ±4.7845E-06	0.697531242 ±9.57609E-07	0.5085 44922	0.5029 29688
C	0.700675771 ±3.0465E-05	0.697837009 ±1.53214E-06	0.5085 44922	0.5029 29688

(A-KL replaced by $\frac{\Delta\mu^2}{2}$; B-Random sampling; C-MCMC sampling, the value after ± is the variance of corresponding value for 100 times samplings)

6. Conclusion

PAC-Bayes risk bound combining Bayesian theory and structure risk minimization for stochastic classifiers has been considered as a framework for deriving some of the tightest generalization bounds. A major issue for calculating the bound is the unknown prior and posterior distributions of the concept space. In this paper, we formulated the concept space as Reproducing Kernel Hilbert Space (RKHS) using the kernel method. 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.

We used two sampling methods to draw examples in concept space with the normal distribution assumption. One of our feature works is to relax this assumption with the consideration of prior distribution of concept space. Another work is to incorporate the feedbacks of sampled concepts to training examples.

Acknowledgement

This work is partly supported by the Natural Science Funding of China under grand number 61170177 and innovation funding of Tianjin University.

References

[1] L. Bottou and O. Bousquet, "13 The Tradeoffs of Large-Scale Learning," *Optimization for Machine Learning*, 2011.

[2] S. Colton, A. Pease, and J. Charnley, "Computational creativity theory: The FACE and IDEA descriptive models," in *2nd International Conference on Computational Creativity*, 2011.

[3] L. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. **27**, no. 1134-1142, 1984.

[4] D. McAllester, "Some pac-bayesian theorems," *Some pac-bayesian theorems*, vol. **37**, no. 3, pp. 355–363, 1998.

[5] J. Langford, "Tutorial on practical prediction theory for classification," *Journal of Machine Learning Research*, vol. **6**, pp. 273–306, 2005.

[6] E. Morvant, S. K. Co, and L. Ralaivola, "PAC-Bayesian Generalization Bound on Confusion Matrix for Multi-Class Classification," in *The 29th International Conference on Machine Learning*, Edinburgh, Scotland, 2012, pp. 1–13.

[7] G. Lever and F. Laviolette, "Distribution-Dependent PAC-Bayes Priors," in *21st International conference on Algorithmic learning theory*, 2010, pp. 119–133.

[8] Y. Seldin and N. Tishby, "A PAC-Bayesian approach to unsupervised learning with application to co-clustering analysis," *Journal of Machine Learning Research*, pp. 1–46, 2010.

[9] J. Shawe-Taylor and D. Hardoon, "PAC-Bayes Analysis Of Maximum Entropy Learning," *12th International Conference on Artificial Intelligence and Statistics*, 2008.

[10] M. Marchand and M. Shah, "PAC-Bayes Learning of Conjunctions and Classification of Gene-Expression Data," *Advances in Neural Information Processing Systems*, no. 17, 2004.

[11] Y. Seldin, "A PAC-Bayesian Analysis of Co-clustering, Graph Clustering, and Pairwise Clustering," in *ICML 2010 Workshop on Social Analytics: Learning from human interactions*, 2010, pp. 1–5.

[12] M. Higgs and J. Shawe-Taylor, "A PAC-Bayes bound for tailored density estimation," in *21st international conference on Algorithmic learning theory*, vol. **6331**/2010, 2010, pp. 148–162.

[13] L. Ralaivola, "Chromatic pac-bayes bounds for non-iid data," *Journal of Machine Learning Research*, vol. **5**, pp. 416–423, 2009.

Li Tang is a PhD Candidate in the School of Computer Science and Technology of Tianjin University and a lecturer in information science and technology Department of Tianjin University of Finance and Economics. She received her master degree majored in Computer Application from Tianjin University of Finance and Economics. She has published several research papers indexed by EI. Her research interests include machine learning and data mining.

Hua Yu is a PhD Candidate and an Engineer in the School of Computer Science and Technology of Tianjin University. She received her master degree majored in geography information system from Shandong University of Science and Technology. She has published several research papers indexed by SCI and EI. Her research interests include machine learning, data mining and semantic web.

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. Then, he worked at the National University of Singapore and the Institute for Infocomm Research (I^2R) 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 etc. Dr. Gong's research interests include Bayesian learning theory, machine learning, distributed information system and bioinformatics.