A Sparse Gaussian Processes Classification Framework for Fast Tag Suggestions

Yang Song
Computer Science and Engineering,
Pennsylvania State University,
University Park, PA 16802, USA
yasong@cse.psu.edu

Lu Zhang
Department of Statistics,
Pennsylvania State University,
University Park, PA 16802, USA
lzz114@psu.edu

C. Lee Giles
Information Sciences and Technology,
Pennsylvania State University,
University Park, PA 16802, USA
giles@ist.psu.edu

ABSTRACT

Tagged data is rapidly becoming more available on the World Wide Web. Web sites which populate tagging services offer a good way for Internet users to share their knowledge. An interesting problem is how to make tag suggestions when a new resource becomes available. In this paper, we address the issue of *efficient* tag suggestion. We first propose a multi-class sparse Gaussian process classification framework (SGPS) which is capable of classifying data with very few training instances. We suggest a novel prototype selection algorithm to select the best subset of points for model learning. The framework is then extended to a novel multi-class multi-label classification algorithm (MMSG) that transforms tag suggestion into the problem of multi-label ranking. Experiments on bench-mark data sets and real-world data from Del.icio.us and BibSonomy suggest that our model can greatly improve the performance of tag suggestions when compared to the state-of-the-art. Overall, our model requires *linear* time to train and *constant* time to predict per case. The memory consumption is also significantly less than traditional batch learning algorithms such as SVMs. In addition, results on tagging digital data also demonstrate that our model is capable of recommending relevant tags to images and videos by using their surrounding textual information.

Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering—*algorithms*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Performance

Keywords

tagging system, gaussian processes, prototype selection, multi-label classification

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

1. INTRODUCTION

The amount of web resources and data continues to grow with the emergence of Web 2.0 sites. As an example, del.icio.us[1] and Flickr[2] have attracted a significant amount of Internet traffic, as well as millions of Internet users. Recent statistics indicated that del.icio.us gets roughly 150,000 posts per day while Flickr gets 1,000,000 photos per day. These web sites allow users to specify keywords or tags for resources, which in turn facilitates the organizing and sharing of these resources with other users. Since the amount of tagged data potentially available is virtually free and unlimited, interest has emerged in investigating the use of data mining and machine learning methods for automated tag suggestions, otherwise known as social annotation [1, 3, 7, 20].

Existing approaches to tag suggestion can be roughly classified into two categories. By mining usage patterns from current users, *collaborative filtering* can be applied to suggest tags from users who share similar tagging behaviors [3, 7]. This approach requires a look-up table of between-user similarities, which is usually in the form of weighted matrices that are computed and stored in advance. Another approach is for the semantic meanings between tags to be leveraged to improve user experience by aggregating similar tags into clusters as in [1, 20]. Here, tag co-occurrence and tag distributions are explicitly taken into account as measurements of tag similarities. The number of optimal clusters are usually tuned manually.

Unfortunately, while both *effectiveness* and *efficiency* need to be addressed for ensuring the performance of the tagging services, most of the existing work has focused on effectiveness [1, 3, 7]. Efficiency, while not being totally ignored, has only been of recent interest [20].

In this paper, we address the efficiency of tag suggestion from a machine learning perspective. We propose a novel sparse Gaussian processes (GP) framework for suggesting multiple tags simultaneously. The proposed model can be fit into any kind of multi-class classification tasks with the advantage of computation efficiency in both the training and test stages. Generally, our model consumes a very small amount of memory and takes linear time proportional to the number of samples for training. The prediction time is constant per case. The performance on tagged data indicates a better precision and less computational cost than the state-of-the-art [20].

[1]http://del.icio.us/
[2]http://www.flickr.com/

1.1 Why Gaussian Processes for Tagging?

The reason of advocating GP for tag suggestion is multifold. First, GP have become an important non-parametric tool for classification. Unlike *generative* classifiers such like Naive Bayes, GP make no assumption on the form of class-conditional density of the data making it immune to any poor performance caused by a false model assumption. Another advantage of GP is that the predicted result of the model yields a probabilistic interpretation, while traditional *discriminative* classifiers such like SVMs usually do not consider the predictive variance of test cases[3]. For tag suggestion where the tagged data (e.g., web pages) usually does not contain any class labels, the user-assigned tags can be used as labels. In this case, GP classifiers that inherit some level of uncertainty can provide a probabilistic classification which tolerates the limitations and possible errors caused by the tags. The predictive variance also offers flexibility of making predictions to new instances.

The other characteristic of tagged data is the unbounded vocabulary. Research has shown a constant growth of the tag vocabulary for a major social bookmarking site CiteULike[4] [5]. Consequently, the tagged data sets used for empirical analysis are usually of high-dimensionality and sparseness [20]. In this case, the efficiency of the model training should also be considered in addition to the performance issue. Nevertheless, massive training data often requires large memory and high computational cost for most discriminative approachs including SVMs. Ad-hoc methods have been developed to select subset for training but those approaches are somewhat heuristic and often performed outside of the model itself. Instead, the sparse GP framework we developed directly selects a subset of most informative documents from all tagged data during training. The prototype selection algorithm we developed requires no extra cost because it reuses the covariance function developed by the GP framework. Consequently, the GP model shows a very promising performance when limited training resources are available by comparing to SVMs. Details can be found in Section 4.

1.2 Our Contributions

Our contributions consist of both new methodology and application to tag suggestion. Our major contributions are:

1. **A novel prototype selection method for multi-class GP classification**. In the literature of GP [19], optimizing the parameters and finding the best subset of points have been performed jointly. However, we propose a prototype selection method to select the most informative points independently. Empirical results show that it performs better in the case of multi-class classification.

2. **A novel multi-label method for tag suggestion**. The proposed multi-class GP framework is naturally extended to address the *multi-label* problem. We treat each associated tag as a label for the document and perform an effective label ranking algorithm for tag suggestion.

3. **Comparison of classification with Support Vector Machines**. SVMs have been well-recognized as one of the best methods for classification. Yet research has shown a close correspondence between SVMs and GP [15]. Empirically, we investigate their performance when there exists very few training instances and conclude that our GP framework performs better in this case.

4. **Comparison to other tag suggestion methods**. We compare our framework with a most recent tag suggestion method [20] as well as SVMs on several real-world data sets and find improved tag suggestion performance.

[3]Though Platt suggested an ad-hoc prob. SVM in [14], it doesn't consider the predictive variance of the function.
[4]http://www.citeulike.org/

2. RELATED WORK

In this section, we briefly review the recent development of tag suggestion methods, followed by the review of the basic Gaussian processes framework.

2.1 Machine Learning for Tag Suggestion

Though some still question the use of computer algorithms to generate tags for social bookmarking services [8], a number of machine learning frameworks have been proposed to address the problem of automatic tag recommendation for both text and digital data on the web [3, 1, 12, 20]. Recent work has shown the effectiveness of leveraging user tags to improve language models [22].

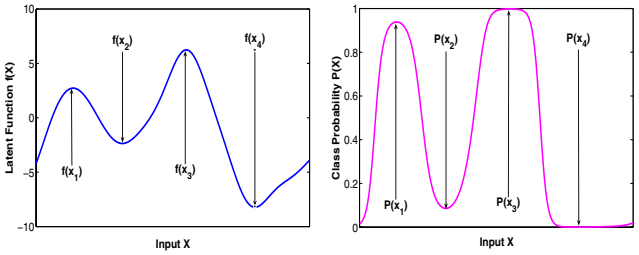
In [3], the authors suggested a method named P-TAG for automatically generating personalized tags in a semantic fashion. They paid particular attention to personalized annotations of web pages. In a document-oriented approach, a web page is compared with a desktop document using either cosine similarity or latent semantic analysis. Keywords are then extracted from similar documents for recommendation. The second keyword-oriented approach alternatively finds the co-occurrence of terms in different documents and recommends the remaining tags from similar desktop documents to the web page. The third hybrid approach combines the previous two methods. From a collaborative filtering point of view, the first two methods can be interpreted as item-based CF with the item being documents and keywords respectively. Their methods, however, do not investigate the behaviors between different users for similar web pages.

A clustering-based approach was proposed in [1] to aggregate semantically related user tags in to similar clusters. Tags are represented as graphs where each node is a tag and the edge between two nodes corresponds to their co-occurrence in the same documents. Tags in the same cluster were recommended to the users based on their similarities. Similarly, an automatic annotation method for images was proposed in [12]. A generative model is trained by exploiting the statistical relationships between words and images. A discrete distribution (D2-) clustering algorithm was introduced for prototype-based clustering of images and words, resulting in a very efficient model for image tagging.

Recently, a method that combines the power of clustering and mixture models was proposed in [20]. The annotated documents were first represented into a triplet of (words, docs, tags) by two bipartite graphs, which were then clustered into topics by a spectral recursive embedding (SRE) [23]. To deal with the sparseness of the topic clusters, a two-way Poisson mixture model (PMM) [13] was applied to simultaneously group documents into components and cluster words. Inference for new documents was based on the posterior probability of topic distributions. Tags were recommended according to the within-cluster tag rankings.

2.2 Gaussian Process Classification

A Gaussian process (GP) is a *stochastic* process consists of a collection of random variables $\mathbf{x}$, which forms a multi-



(a) (b)

Figure 1: One-dimensional illustration of Gaussian process construction for classification. (a) A latent function $f(X)$ drawn from Gaussian Process, where $f(x_i)$ denotes the latent function value of point x_i. (b) The class probability of X after scaling $f(X)$ into $(0,1)$ by a sigmoid function $\Phi(f_i) = 1+\exp(-f_i)^{-1}$, where $P(x_i)$ denotes the class probability at x_i.

variate Gaussian distribution specified by a mean function $\mu(\mathbf{x})$ and covariance function $k(\mathbf{x},\mathbf{x}')$. For classification, the objective is to assign a new observation $\mathbf{x}_*$ to one or more predefined classes denoted by $y_* \in \{1,...,C\}$. GPs can not be applied to the classification task directly because the values of y are not continuous. Consequently, a *latent function* $f(\mathbf{x})$ is employed to infer the labels. The GP prior is therefore placed over $f(\mathbf{x})$. Fig 1 (a) illustrates an one-dimensional case of the latent function with mean 0. To make a prediction given a new $\mathbf{x}_*$, one first determine the predictive distribution $p(\mathbf{f}_*|\mathbf{f})$, where $\mathbf{f}$ is obtained from the training set, $\mathbf{f}|X_{train} \sim \mathcal{N}(\mathbf{0},\mathbf{K})$, with $\mathbf{K}$ denoting the multivariate covariance matrix. The class probability y_* is then related to the latent function $\mathbf{f}_*$.

3. TRADITIONAL MULTI-CLASS GP MODEL

Denote a training data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1,...,N\}$ with N training points $X = \{\mathbf{x}_i|i = 1,...,N\}$ drawn independent and identically distributed (i.i.d.) from an unknown distribution, and the associated labels $\mathbf{y} = \{y_i|i = 1,...,N\}$, where each point $\mathbf{x}_i$ is a D dimensional feature vector, $\mathbf{x}_i \in \mathbb{R}^D$ and $y_i \in \{1,...,C\}$. Following the convention in [15], we introduce a vector of latent function values of N training points for C classes, which has length CN

$$\mathbf{f} = (f_1^1,...,f_N^1,...,f_1^j,...,f_N^j,...,f_1^C,...,f_N^C)^T, \quad (1)$$

where $\mathbf{x}_i$ has C latent functions $\mathbf{f}_i = (f_i^1,...,f_i^C)$. We further assume that the GP prior over $\mathbf{f}$ has the form $\mathbf{f}|X \sim \mathcal{N}(\mathbf{0},\mathbf{K})$, where $\mathbf{K}$ represents the covariance matrix which is constructed from a pair-wise covariance function $\mathbf{K}(\mathbf{x}_n,\mathbf{x}_{n'}) \triangleq [\mathbf{K}_N]_{nn'}$. Specifically, $\mathbf{K}$ is block diagonal of size $CN \times CN$ in the matrices $\mathbf{K}_1,...,\mathbf{K}_C$, where each $\mathbf{K}_j$ represents the correlations of the latent function values within class j. A wide range of covariance functions can be chosen for GP classification [15].A commonly used function in the classification case is the *squared exponential* function, defined as:

$$[\mathbf{K}_N]_{nn'} = l\exp\left(-\frac{1}{2}\frac{\sum_{d'=1}^{D}\left(x_n^{(d')} - x_{n'}^{(d')}\right)^2}{\mathbf{\Sigma}^2}\right), \quad (2)$$

where $\theta = \{l, \mathbf{\Sigma}^2\}$ corresponds to the *hyper-parameters*.Given the training set $\mathcal{D}$, we can compute the posterior of the latent function by plugging in the Bayes' rule,

$$p(\mathbf{f}|X,y) = \frac{p(\mathbf{f}|x)p(y|\mathbf{f})}{p(X,y)} \overset{i.i.d.}{=} \frac{\mathcal{N}(\mathbf{0},K)}{p(X,y)}\prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{f}_i), \quad (3)$$

which is non-Gaussian. In eq.(3), the conditional probability $p(\mathbf{y}|\mathbf{f})$ has not been decided yet. In the multi-class case, $\mathbf{y}$ is a vector of the length CN (which is the same as $\mathbf{f}$), which for each $i = 1,...,N$ has an entry of 1 for the class which corresponds to the label of the point $\mathbf{x}_i$ and 0 for the rest $C-1$ entries. One of the choices is a *softmax* function:

$$p(y_i^c|\mathbf{f}_i) = \frac{\exp(f_i^c)}{\sum_{c'}\exp(f_i^{c'})}. \quad (4)$$

To proceed, we compute the predictive distribution of the class probability given a new $\mathbf{x}_*$ in two steps. First, compute the latent value $\mathbf{f}_*$ by integrating out $\mathbf{f}$:

$$p(\mathbf{f}_*|X,y,\mathbf{x}_*) = \int p(\mathbf{f}_*|\mathbf{f},X,\mathbf{x}_*)\underbrace{p(\mathbf{f}|X,y)}_{\text{eq.(3)}}d\mathbf{f}, \quad (5)$$

then $\mathbf{y}_*$ can be computed by integrating out $\mathbf{f}_*$:

$$p(\mathbf{y}_*|X,y,\mathbf{x}_*) = \int p(\mathbf{y}_*|\mathbf{f}_*)\underbrace{p(\mathbf{f}_*|X,y,\mathbf{x}_*)}_{\text{eq.(5)}}d\mathbf{f}_*. \quad (6)$$

This method takes $O(N^3)$ to train due to the inversion of the covariance matrix $\mathbf{K}$. A range of *sparse* GP approximations have been proposed [11, 16]. Most of these methods seek a subset of M ($M \ll N$) training points which are *informative* enough to represent the entire training set. Consequently, the training cost is reduces to $O(NM^2)$ and the corresponding test cost to $O(M^2)$. Next we discuss a sparse way to reduce the computational cost in the multi-class case.

4. OUR MULTI-CLASS SPARSE GP MODEL

Our model involves several steps. First, we choose M ($M \ll N$) points (denote as $\bar{X} = \{\bar{\mathbf{x}}_m\}_{m=1}^M$) from the training set. Then we generate their latent functions $\bar{\mathbf{f}}$ from the prior. The corresponding $\mathbf{f}$ for the entire training set is thus drawn conditionally from $\bar{\mathbf{f}}$. See Figure 2 for details.

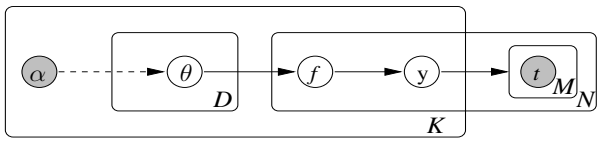


Figure 2: Graphical representation of our sparse multi-class GP model. θ is the hyper-parameter that define the latent function f. α denotes the extra parameter for placing a distribution over θ.

First, assume that the M points have already been chosen. Then place a GP prior on $\bar{X}$, which uses the same covariance function as shown in eq. (2), such that these points have a similar distribution to the training data,

$$p(\bar{\mathbf{f}}|\bar{X}) = \mathcal{N}(\bar{\mathbf{f}}|\mathbf{0},\mathbf{K}_M). \quad (7)$$

Given a new $\mathbf{x}_*$, we utilize M latent functions $\bar{\mathbf{f}}$ for prediction. We compute the latent values $\mathbf{f}_*$ by integrating the likelihood with the posterior:

$$p(\mathbf{f}_*|\mathbf{x}_*, X, \mathbf{y}, \bar{\mathbf{f}}, \bar{X}) = \int \underbrace{p(\mathbf{f}_*|\mathbf{x}_*, \bar{\mathbf{f}}, \bar{X})}_{A} \underbrace{p(\bar{\mathbf{f}}|X, \mathbf{y}, \bar{X})}_{B} d\bar{\mathbf{f}}, \quad (8)$$

where A represents the single data likelihood by applying to the reduced set of points. With $\bar{\mathbf{f}}$ determined, the likelihood can be treated as a bivariate normal distribution, which follows a normal distribution:

$$\mathbf{f}_*|\mathbf{x}_*, \bar{\mathbf{f}}, \bar{X} \sim \mathcal{N}(\mathbf{f}_*|\mathbf{k}_{\mathbf{x}_*}^T \mathbf{K}_M^{-1} \bar{\mathbf{f}}, K_{\mathbf{x}_*\mathbf{x}_*} - \mathbf{k}_{\mathbf{x}_*}^T \mathbf{K}_M^{-1} \mathbf{k}_{\mathbf{x}_*}), \quad (9)$$

where $\mathbf{k}_{\mathbf{x}_*} = \mathbf{K}(\bar{\mathbf{x}}, \mathbf{x}_*)$ and $[\mathbf{K}_M]_{ij} = \mathbf{K}(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$.

Nevertheless, the problematic form of posterior B does not follow a normal distribution and has to be approximated.

4.1 Laplace Approximation for the Posterior

Our method to approximate B in eq.(8) is based on the Laplace approximation, which were used in [15] for binary classification. Using the Bayes' rule,

$$\begin{aligned} p(\bar{\mathbf{f}}|X, \mathbf{y}, \bar{X}) &= \frac{p(\bar{\mathbf{f}}|\bar{X}) p(\mathbf{y}|\bar{\mathbf{f}}, X, \bar{X})}{p(\mathbf{y}|X, \bar{X})} \\ &= \frac{p(\bar{\mathbf{f}}|\bar{X}) \overbrace{\int p(\mathbf{f}|\bar{\mathbf{f}}, X, \bar{X}) p(\mathbf{y}|\mathbf{f})\, d\mathbf{f}}^{C}}{p(\mathbf{y}|X, \bar{X})}. \quad (10) \end{aligned}$$

The detail of the derivation is long and omitted. The approximated mean and variance of eq.(10) is:

$$\begin{aligned} \boldsymbol{\mu}_* &\simeq \boldsymbol{\mu}_p, \\ \boldsymbol{\Sigma}_* &= \mathbf{K}_M + \boldsymbol{\Sigma}_p. \\ \text{where } \boldsymbol{\mu}_p &= \frac{\mathbf{Q}^{-1}\mathbf{P}}{2}, \boldsymbol{\Sigma}_p = \mathbf{Q}^{-1}, \\ \mathbf{Q} &= (\mathbf{K}_{NM}\mathbf{K}_M^{-1})^T \boldsymbol{\Lambda}^{-1} (\mathbf{K}_{NM}\mathbf{K}_M^{-1}) + \mathbf{K}_M, \\ \mathbf{P} &= \hat{\mathbf{f}}^T \boldsymbol{\Lambda}^{-1} (\mathbf{K}_{NM}\mathbf{K}_M^{-1}). \quad (11) \end{aligned}$$

4.1.1 Determine the class label of test documents

The final step is to assign a class label to the observation $\mathbf{x}_*$, given the predictive class probabilities by integrating out the latent function f_*:

$$p(\mathbf{y}_*|\mathbf{x}_*, X, \mathbf{y}, \bar{\mathbf{f}}, \bar{X}) = \int \tilde{p}(\mathbf{f}_*|\mathbf{x}_*, X, \mathbf{y}, \bar{\mathbf{f}}, \bar{X}) p(\mathbf{y}_*|\mathbf{f}_*) df_*, \quad (12)$$

which again cannot be solved analytically. One way to approximate is to use cumulative Gaussian likelihood. In [15], the authors estimated the mean prediction by drawing S samples from the Gaussian $p(\mathbf{f}_*|\mathbf{y})$, softmax and averaging the results. Once the predictive distribution of the class probability is determined, the final label of $\mathbf{x}_*$ can be decided by choosing the maximum posterior (MAP):

$$t(\mathbf{x}_*) = \arg\max_c p(y(\mathbf{x}_*)^c|\cdot), \quad c = 1, ..., C. \quad (13)$$

4.2 Informative Points Selection

It remains to optimize the parameters $\Theta = \{\theta, \bar{X}\}$, which contain the hyper-parameters $(l, \boldsymbol{\Sigma})$ for the covariance matrix $\mathbf{K}$ as well as finding the subset $\bar{X}$ of M points. Traditionally, they are optimized jointly by optimizing the marginal likelihood of the training data. In our approach, we instead treat them individually.

4.2.1 Parameter Inference for the Covariance Matrix

The marginal likelihood of $\mathbf{y}$ can be obtained by integrating out $\bar{\mathbf{f}}$,

$$p(\mathbf{y}|X, \bar{X}, \Theta) = \int p(\mathbf{y}|X, \bar{X}, \bar{\mathbf{f}}) p(\bar{\mathbf{f}}|\bar{X}) d\bar{\mathbf{f}} = \int \exp(\mathcal{L}(\bar{\mathbf{f}})) d\bar{\mathbf{f}}. \quad (14)$$

With a Taylor expansion of $\mathcal{L}(\bar{\mathbf{f}})$ around $\hat{\bar{\mathbf{f}}}$ we find

$$\mathcal{L}(\bar{\mathbf{f}}) \simeq \mathcal{L}(\hat{\bar{\mathbf{f}}}) + \underbrace{(\bar{\mathbf{f}} - \hat{\bar{\mathbf{f}}}) \nabla_{\bar{\mathbf{f}}} \mathcal{L}(\hat{\bar{\mathbf{f}}})}_{=0} + \frac{1}{2} (\bar{\mathbf{f}} - \hat{\bar{\mathbf{f}}})^T \nabla\nabla_{\bar{\mathbf{f}}} \mathcal{L}(\bar{\mathbf{f}}) (\bar{\mathbf{f}} - \hat{\bar{\mathbf{f}}}).$$

Therefore, the approximation of the marginal likelihood can be written as

$$\begin{aligned} &p(\mathbf{y}|X, \bar{X}, \Theta) = \\ &\exp(\mathcal{L}(\hat{\bar{\mathbf{f}}})) \int \exp\left(\frac{1}{2} (\bar{\mathbf{f}} - \hat{\bar{\mathbf{f}}})^T \nabla\nabla_{\bar{\mathbf{f}}} \mathcal{L}(\bar{\mathbf{f}}) (\bar{\mathbf{f}} - \hat{\bar{\mathbf{f}}}) \right) d\bar{\mathbf{f}}. \quad (15) \end{aligned}$$

The log marginal likelihood can be obtained by taking logarithm on both sizes of the above equation,

$$\log p(\mathbf{y}|X, \bar{X}, \Theta) = \mathcal{L}(\hat{\bar{\mathbf{f}}}) - \frac{CN}{2} \log 2\pi - \frac{1}{2} \log |\nabla\nabla_{\bar{\mathbf{f}}} \mathcal{L}(\bar{\mathbf{f}})|, \quad (16)$$

which can be maximized w.r.t. the parameters Θ to obtain $\hat{l}$ and $\hat{\boldsymbol{\Sigma}}$. Note that each $\boldsymbol{\Sigma}_c$ is a $D \times D$ symmetric matrix, where D is the number of dimensions. We assume that each dimension is independent, thus simplifies $\boldsymbol{\Sigma}_c$ to be a diagonal matrix. However, this still yields DC parameters to estimate for $\boldsymbol{\Sigma}$. Therefore, we further assume that within each class c, the covariance of each dimension is the same, so that the total number of parameters for $\boldsymbol{\Sigma}_c$ is reduced to C.

4.2.2 Prototype selection for $\bar{X}$

The original gradient calculation in eq.(16) is very complicated. However, we can simplify it with the assumption made on the covariance matrix. Since each $\boldsymbol{\Sigma}_c$ is now independent of each other, we can estimate the locations of the active points regardless of the choices of l and $\boldsymbol{\Sigma}$. We greedily find the locations of $\bar{X}$ by stochastic gradient descent method. This is similar to finding the optimal *prototypes* for each class, which is a subset of points that contains enough information for each class. Our method for optimal prototype search is parallel to [18], which is used for K-nearest neighbor classification. We select a set of M prototypes by minimizing the misclassification rate of the training set,

$$\mathfrak{L}(X, \bar{X}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{m=1}^{M} P(\bar{\mathbf{x}}_m|\mathbf{x}_n)(1 - \mathbb{I}(\bar{y}_m = y_n)), \quad (17)$$

where the indicator function $\mathbb{I}$ is 1 if the condition is hold and 0 otherwise. The likelihood $P(\bar{\mathbf{x}}_m|\mathbf{x})$ can be calculated by plugging in the normalized covariance:

$$P(\bar{\mathbf{x}}_m|\mathbf{x}) = \frac{\mathbf{k}_{\bar{\mathbf{x}}_m \mathbf{x}}}{\sum_{m'=1}^{M} \mathbf{k}_{\bar{\mathbf{x}}_{m'} \mathbf{x}}}. \quad (18)$$

We can further rewrite the loss function in eq.(17) by removing the indicator function:

$$\mathfrak{L}(X, \bar{X}) = \frac{1}{N} \sum_{n} \underbrace{\sum_{\{m: \bar{y}_m \neq y_n\}} P(\bar{\mathbf{x}}_m|\mathbf{x}_n)}_{\ell_m}, \quad (19)$$

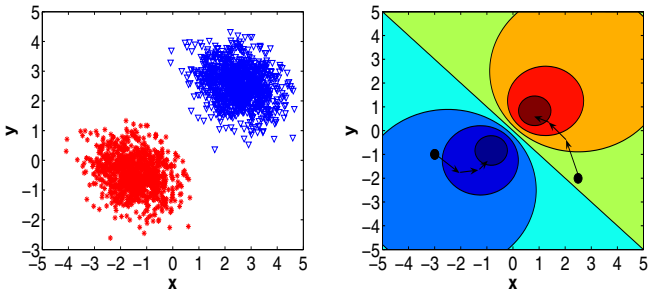


Figure 3: An example of prototype selection with $M = 2$. Left figure shows the original distribution; right figure, contour-plots the results of descent where black dots are the starting points.

where $\mathfrak{l}_m$ indicates the individual cost of misclassification, which is continuous in the interval $(0, 1)$. Therefore, it can be minimized by gradient descent w.r.t. $\bar{X}$,

$$\begin{aligned} & \bar{\mathbf{x}}_m(t+1) \\ = \; & \bar{\mathbf{x}}_m(t) - \alpha(t)\nabla_{\bar{\mathbf{x}}_m}\mathfrak{l}_m(t) \\ = \; & \bar{\mathbf{x}}_m(t) + \alpha(t)p(\bar{\mathbf{x}}_m|\mathbf{x})\left(\mathbb{I}(\bar{y}_m \neq y_n) - \mathfrak{l}_m(t)\right)\frac{\delta \mathbf{k}_{\bar{\mathbf{x}}_m \mathbf{x}}}{\delta \bar{\mathbf{x}}_m}. \\ = \; & \bar{\mathbf{x}}_m(t) + \begin{cases} \mathfrak{l}_m(1-\mathfrak{l}_m)P(\bar{\mathbf{x}}_m|\mathbf{x})(\bar{\mathbf{x}}_m - \mathbf{x}) & \text{if } \bar{y}_m \neq y_n \\ -\mathfrak{l}_m(1-\mathfrak{l}_m)P(\bar{\mathbf{x}}_m|\mathbf{x})(\bar{\mathbf{x}}_m - \mathbf{x}) & \text{otherwise} \end{cases} \end{aligned}$$

Here $\alpha(t) > 0$ is a small enough number which specifies the step length of the descent. The program stops when a stopping criterion is reached. We further notice that only those points falling into a particular area of the input space can contribute to the update of the prototypes. This fact is explained as the *window* rule in [9]. So we can speed up the prototype updates by searching over those points only. Figure 3 shows an example of two prototypes. It can be seen that after three steps of descent, our algorithm successfully finds informative points for each class.

For brevity we hyphenate our method as Sparse Gaussian process with Prototype Selection (SGPS).

4.3 Discussion of the Computational Cost

In our sparse framework, only the covariance matrix $\mathbf{K}_M$ for the M prototypes is required to be inverted. To be exact, $\mathbf{K}_M$ needs to be inverted when calculating $\mathbf{\Lambda}$, $\mathbf{f}'$, $\mathbf{Q}$ and $\mathbf{P}$ in eq.(11). For efficiency, Cholesky decomposition is often employed, which ensures that for N training points distributed in C classes, the training stage can be realized in $O(M^2NC)$ time,with $O(M^2C)$ per prediction.

As for the cost of prototype selection, since the updates reuses covariance matrix in eq.(18), no additional storage and computation are required. Therefore, the gradient descent can be efficiently updated in at most $O(NC)$ time.

5. MULTI-LABEL TAG SUGGESTION

So far, we have only considered the case that the each observation is single-labeled, i.e., belongs to only one class. In fact, many real-world problems are multi-labeled. In the case of tagged data, each tag associated with a document may be treated as a label, which may or may not refer to the same topic as other labels. Thus, the problem of tag suggestion can be transformed into a multi-label classification problem where the objective is to predict the probability of a document with all possible tags (labels) given a fixed tag vocabulary and associated training documents.

Algorithm 1 Multi-label Multi-class Sparse GP Classification (MMSG) for Tag Suggestion

Input: training data $\mathcal{D} : \{(\mathbf{x}_i, \mathbf{y}_i)\}_1^N, \mathbf{x}_i \in \mathbb{R}^d, \mathbf{y}_i = \{y_{i1}, ..., y_{i\tilde{K}}\}$
M: number of prototypes
$\mathbf{k}$: covariance function
begin training procedure
for $i = 1 : N$
  $c_i = \max(s(\mathbf{y}_i))$ //decide the category of $\mathbf{x}_i$
end for
Train a GP classifier given $\{(\mathbf{x}_i, c_i)_1^N, M, \mathbf{k}\}$
Output: $\bar{X}, \bar{\mathbf{f}}$
begin test procedure
Input: a test object $\mathbf{x}_*$
Decide its category probabilities $\mathbf{c}_*$ given $\bar{X}, \bar{\mathbf{f}}$ (eq.(12))
for each category $m \in \{1, ..., C\}$
  for each label $y_{ij}^{(c)} \in \{y_{i1}^{(c)}, ..., y_{i\tilde{K}}^{(c)}\}$
    $P(y_{ij}^{(c)}|\mathbf{x}_*) = Rank_{y_{ij}}^{(c)} \cdot c_m(\mathbf{x}_*)$
  end for
end for
Output: $P(\mathbf{y}_*, \mathbf{x}_*)$

The problem of multi-label classification (MLC) is arguably more difficult than the traditional single-label classification task, since the number of combinations for two or more classes is exponential to the total number of classes. For N classes, the total number of possible multi-labeled class is 2^N, making it unfeasible to expand from an algorithm for single-label problems. Much research has been devoted to increasing the performance of MLC and generalize the framework to single-label classification; see related work for more information[21].

As pointed out in [2], multi-label classification can be treated as a special case of label ranking, which can be realized if the classifiers provide real-valued confidence scores or a posterior probability estimates for classification outcomes. Thus, the multi-class SGPS model readily maps to this problem, since the output vector $\mathbf{y}_*$ contains real-valued scores of the posterior class probabilities. Specifically, in the multi-label case, we assume that the class label of a training instance $\mathbf{x}_i$ is no longer a binary value, but rather a vector $\mathbf{y}_i$ of binary values where each y_{ij} denotes the existence/absence of $\mathbf{x}_i$ in class j. We further assume that these

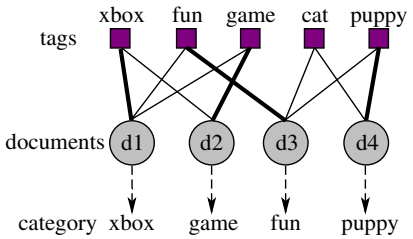


Figure 4: Example of document-tag graph. Each document is associated with multiple tags. Tag with the highest frequency is treated as the category of that document (shown in bold line).

class probabilities can be ranked according to their values, where $s(y_{im}) > s(y_{in})$ indicates that y_{im} is preferred to y_{in}. In the context of tags, the value of a tag is defined as the number of times it has been used to annotate the specific object. So if a document d_1 (cf Figure 4) is tagged 4 times with *game*, 3 times with *fun* and 5 times with *xbox*, we can rearrange the labels in the descending order, yielding, { *xbox*(5), *game*(4), *fun*(3) }. Note that normalization is usually required to ensure the well-defined class probability, thus the class probabilities of the above case become $\{0.42, 0.33, 0.25\}$. Figure 4 shows an example of 4 documents and 5 tags with their categories in bold lines.

In this way we can transform multi-class multi-label classification into *multi-category single-label* classification. Specifically, we first assign each $\mathbf{x}_i$ into a single *category* c which corresponds to its top-ranked label (e.g., in the above case, the category is *xbox*). Each category contains a set of labels that belong to the objects in that category. Intuitively, tags that belong to the same category are more semantically related than tags in different categories, i.e., tags in the same category have a higher co-occurrence rate. However, it should be noted that an individual tag could belong to multiple categories, e.g., in Figure 4, *fun* appears in two categories. The above two phenomenon can be roughly explained by the behavior of *polysemy* and *synonymy* in linguistics. Table 1 shows three ambiguous tags and their corresponding categories in one of our experiments.

tags	categories
apple	mac **apple** computers osx technology IT food health **apple** nutrition fruit green
tiger	photos nature animal **tiger** cute animals sports video **tiger** woods golf games
opera	music art **opera** culture design download software browser **opera** web tools internet

Table 1: Example of ambiguous tags from del.icio.us.

Given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_1^N$, the within-category scores of all possible labels are defined as

$$Rank_{y_{i'}}^{(c)} = \frac{1}{Z^{(c)}} \sum_{i:\mathbf{x}_i \in c} \sum_j s(y_{ij}) \mathbb{I}(y_{ij} = y_{i'}), y_{i'} = \{y_{i1}, ..., y_{i\tilde{K}}\} \tag{20}$$

where $Z^{(c)}$ is a normalization factor for category c. We summarize this approach in Algorithm 1, $\tilde{K}$ refers to the total number of possible labels. During the training phase, we train an SGPS model for C categories, as well as calculating the within-category scores for all labels. In the test phase, we use the model first to determine the probabilistic distribution of the categories given a new test case. Then combine this evidence with the within-category scores of tags in a multiplicative fashion to obtain the final label distribution. The labels are sorted in descending order based on the estimated likelihoods, the top-ranked tags are used for recommendation. Figure 5 illustrates the process.

6. EXPERIMENTS

To assess the performance of the proposed framework, we first evaluate our GP framework SGPS with synthetic and bench-mark data, then measure the quality of the tag suggestion algorithm (MMSG) using real-world data sets.

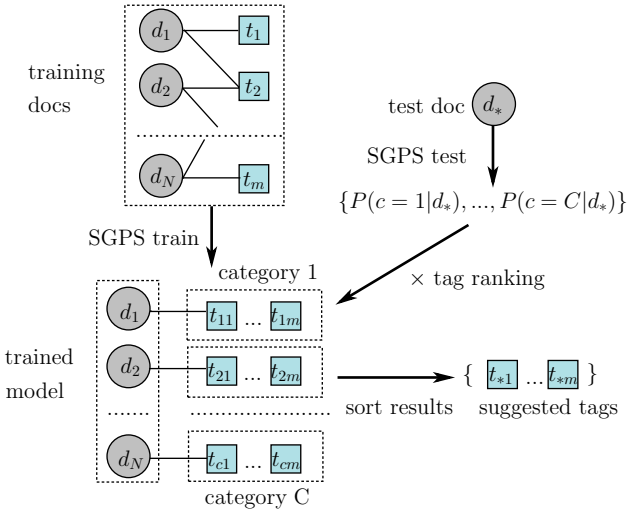


Figure 5: The training and test processes of MMSG. Each d_i is a document and each t_i is a tag.

6.1 Comparison of Sparse Gaussian Models

For comparison, we use the multi-class Laplace approximation algorithm by Williams (GPLA)[15], to see how differently our sparse framework will perform against the full Gaussian model. We also include the sparse variational Bayesian method (VBGP)[6] and the multi-class informative vector machines (IVM)[5] [16] as two of the sparse Gaussian process models. Three common metrics are employed for assessment: the *marginal likelihood* of the training set in eq.(16), the *predictive error* of the test set, and the *algorithm running time*. For the first metric, we fix the number of prototypes M to be 5%. For the other two, we examine the performance variation with the change of M.

6.1.1 A Toy Data Set

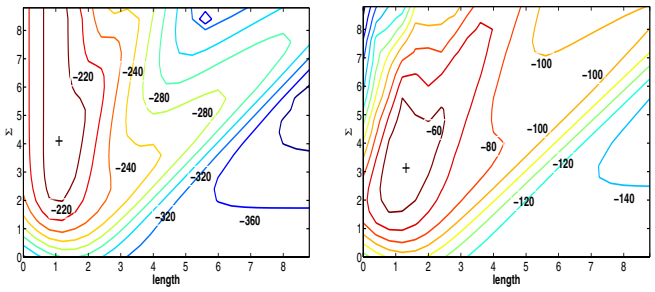
Following [6], we generate a three-class synthetic data set which has ten-dimensions. Only the first two dimensions of the data are responsible for defining the three class labels, while the other eight dimensions are irrelevant. In order to compare with previous work, we use the same experiment setting as in [6]. i.e., we create 500 instances for training, and draw another 5,000 instances independently for testing.

6.1.2 Marginal Likelihood

In order to judge the change of the marginal likelihood, a common approach is to perform a range of experiments with different values of the hyper-parameters. In [10], the authors created a regular 21 × 21 grid for both hyper-parameters. Here, we fix M to be 5% (i.e., 25 training points) and investigate the change of $p(\mathbf{y}|X, \bar{X}, \Theta)$ with $\mathbf{\Sigma}$ and l.

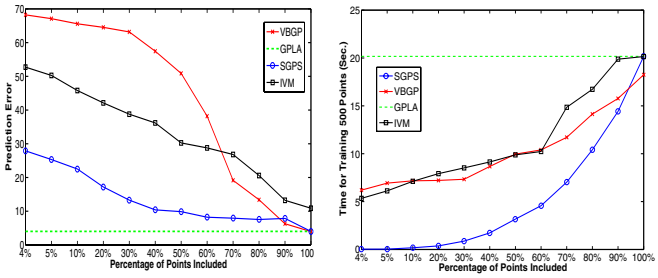
Figure 6 shows a contour plot of the marginal likelihood as a function of l and $\mathbf{\Sigma}$. Usually, higher marginal likelihood yields better predictive performance. In this figure, we notice that the variation of marginal likelihood from our model SGPS preserves the shape of GPLA, with the change of parameters. The result justifies the use of the sparse ap-

[5]Somehow the code of multi-class IVM is not available, we thus use the binary version with one-vs-all classification.



(a) GPLA (b) SGPS

Figure 6: Contour plot of the approximated marginal likelihood, as a function of hyper-parameters l and Σ. The marginal likelihood reaches an optimum at $l = 1.38, \Sigma = 3.13$ for GPLA, and $l = 1.15, \Sigma = 4.05$ for our model SGPS.



(a) Prediction Error. (b) Training Time.

Figure 7: The performance of the toy data set. The full Gaussian model GPLA is used as a baseline. Our model (SGPS) achieves better predictions and requires less running time than VBGP and IVM.

proximation SGPS of the full Gaussian model GPLA for the toy data set.

6.1.3 Prediction Error

Figure 7(a) plots the prediction error of the 5,000 test points for the three methods. Our model SGPS incurs significantly less error with M from 4% to 90%. Especially when M is quite small, our model exhibits its robustness. With only 4% of the training points, our model is capable of making more than 70% correctly predictions, which quickly converges to the performance of the full model GPLA. Whereas for VBGP and IVM, the outcome is not attractive in the lower region of M, although VBGP slightly outperforms (Error = 3.96%) our model (Error = 4.01%) when $M = 100\%$. Overall, our model exhibits better predictive performance.

6.1.4 Computational Cost

The training cost of the three methods are depicted in Figure 7 (b). The test result shows similar patterns thus ignored here. Results are averaged over 10 run with standard deviations eliminated for better interpretation. It is clear to see that our model requires much less time to train compared to VBGP and IVM, which is consistent with the theoretical foundation. Recall that for our sparse approximation, only $O(M^2NC)$ time is required for training, whereas the number

	GPLA	VBGP	IVM	SGPS
Iris		$(C = 3, N = 150)$		
Marginal likelihood	-187.2	-33.1	-36.7	**-31.8**
Predictive error	3.7	5.8	6.4	**5.2**
Training cost	430.9	53.5	76.1	**28.4**
Test cost	204.2	35.3	48.4	**17.1**
Letter		$(C = 10, N = 20000)$		
Marginal likelihood	-476.3	-122.4	-137.2	**-98.1**
Predictive error	4.5	12.5	13.8	**6.8**
Training cost	3854.2	983.2	1658.3	**1032.4**
Test cost	1777.5	542.2	590.2	**472.5**
Pendigits		$(C = 10, N = 10992)$		
Marginal likelihood	-255.3	-78.5	-84.3	**-65.2**
Predictive error	3.8	13.2	10.5	**5.3**
Training cost	1572.3	665.3	853.2	**501.3**
Test cost	865.2	388.4	390.3	**305.2**

Table 2: Performance on three UCI data sets (C: number of classes, N: number of instances).

is $O(M^2NC^3)$ for the multinomial logit Laplace approximation, both with M prototypes. Thus, when the number of classes become larger, SGPS can exhibit better performance.

6.1.5 UCI Multi-class Bench-mark Data

We further assess the model performance using 15 standard multi-class data sets from the UCI ML Repository[6]. Due to space limitations, we only report 3 data sets here. We randomly select 60% of the data for training and the rest 40% for test. The experiment is repeated 50 times and we report the average. The experimental results with means and standard deviations can be found in Table 2.

The predictive error of SGPS is consistently lower than the competitors. Meanwhile, our model requires less time to train and make predictions. For larger data sets (e.g., letter and pendigits), SGPS gains much better performance, which indicates a good scalability in practical uses. We also notice that VBGP often outperforms IVM.

6.2 Application on Multi-label Data Sets

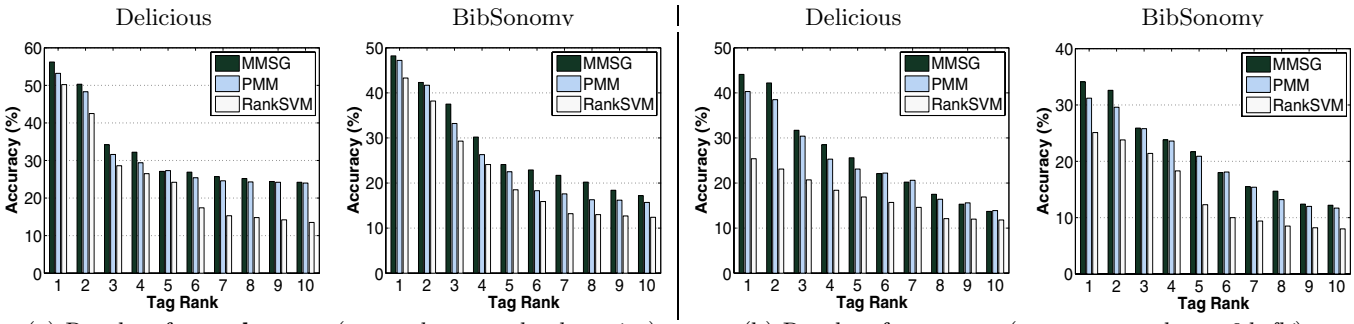
To justify the effectiveness of our multi-label algorithm MMSG, we empirically analyze its performance using both bench-mark and real-world data. We use SVM^{struct} as the multi-label extension of SVM for comparison[7].

6.2.1 Reuters Corpus Volume I (RCV1)

The *Reuters2000* data set is employed as a bench-mark. The data contains 800,000 newswire stories, each of which belongs to a subset of 101 categories. The data is stored in TFIDF format with more than 47,000 features. For our experiment, we restrict the categories to three: *topics, regions* and *industries*. Four standard metrics are used: *Precision, Rank Loss, Hamming Loss* and *One Error* [4]. For precision, higher percentage indicates better performance, while for the other three, low scores are generally better.

Table 3 shows that our algorithm has lower performance for all four metrics but is computationally much more efficient. A reduction of 13% of the precision required almost 80% less memory. The program run time is reduced from

[6]http://mlearn.ics.uci.edu/MLRepository.html
[7]http://svmlight.joachims.org/



(a) Results of **popular** tags (e.g, *web, network, clustering*) (b) Results of **rare** tags (e.g., *asp.net, latex, 3d, fbi*)

Figure 8: Tag suggestion results on popular tags and rare tags for Delicious and BibSonomy. MMSG uses only $M = 5\%$ prototypes and outperforms PMM and SVM.

107.3 minutes to 32.5 minutes. We also trained a sparse SVM (denoted as S-S) by using 5% of the data. The result shown here is apparently worse than MMSG.

	SVM	MMSG	S-S
Precision	0.69	0.59	0.48
R-Loss	0.26	0.32	0.38
H-Loss	0.25	0.27	0.33
O- Error	0.20	0.30	0.35

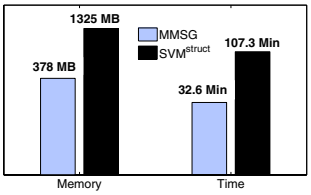


Table 3: Multi-label classification results on RCV1. For SVM^{struct}, L_1-loss function is used. MMSG and S-S use $M = 5\%$ points for training.

del.icio.us		BibSonomy	
Tag Name	**Frequency**	**Tag Name**	**Frequency**
internet	1743	tools	2459
technology	1543	computing	2294
java	1522	software	1974
software	1473	blog	1717
web	1429	internet	1647
photography	1375	web	1631
news	1328	analysis	1562
music	1291	data	1248
business	1115	search	1196
travel	1092	design	1117

Table 4: Top 10 most popular tags in del.icio.us and BibSonomy with respective frequencies.

6.3 Experiments on Tag Recommendation

We compare the performance of tag recommendation of our algorithm with two other approaches. We first use SVM^{struct} to train a multi-label model and use the same ranking function as in eq.(20) to return top ranked tags. We also compare to the tag suggestion method using the Poisson mixture model (PMM) [20], which often outperforms other existing tag suggestion methods.

6.3.1 Del.icio.us and BibSonomy

We collected data from del.icio.us and BibSonomy[8] between Oct 15 2007 and Jan 10 2008. For both data sets, we randomly sampled 50 tags from the tag lists. For each tag, we retrieved the content of bookmarks with related tags. A typical instance in our experiment contains title, content and tags. The titles and contents are treated with equal importance here[9]. Overall, the del.icio.us data contains 45,715 unique items with 93,215 words, while BibSonomy has 14,200 items with 37,605 words. The total numbers of tags are 8,792 and 6,321, respectively. Table 4 shows top 10 tags for both data sets.For training, the data is organized into 50 classes. Considering the temporal characteristics of tags, we ordered the data by time and used the earlier data for training and tested on later data. We performed experiments with training data from 10% to 90%. Due to space limitation, we only report the results of 50% splitting here.

[8]http://www.bibsonomy.org

[9]We performed test by assigning higher weights to titles, with almost same results found.

We present a summary of the experimental results in Table 5. Overall, our model is able to boost the tagging performance when comparing with PMM and SVM, by roughly 6% and 12% on average. Note that this is efficiently achieved by using only 5% of the training instances.

In addition, we also examined the performance of individual tags by looking at the top 10 suggested tags. We are interested in the difference in performance between popular tags (e.g., web, network, clustering) and rare tags (e.g., asp.net, latex, 3d). For each data set, we chose the top-5 most/least popular tags and averaged the suggesting results. Figure 8 depicts the results. It can be observed that MMSG outperforms PMM and SVM in most cases. The top-most suggested tag for del.ici.ous and BibSonomy from MMSG

Algorithm	Precision	Recall	F-Score
delicious			
SVM	40.21%	61.44%	50.63%
PMM	43.52%	62.31%	52.77%
MMSG	47.38%	66.16%	54.23%
BibSonomy			
SVM	33.45%	52.93%	45.56%
PMM	35.21%	55.72%	47.23%
MMSG	39.45%	57.01%	52.32%

Table 5: Tagging performance.

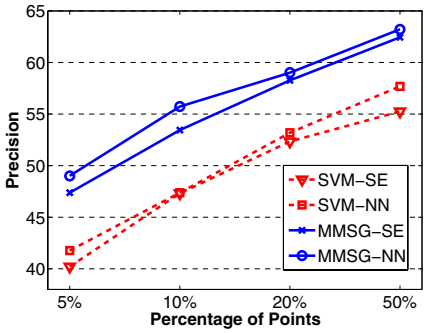


Figure 9: Comparison of tagging performance of SVM and MMSG. Two covariance functions used: SE = squared exponential, NN = neural network.

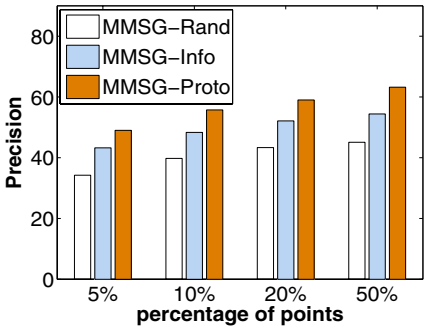


Figure 10: Tagging performance of three selection algorithms. Rand = random selection, Info = information gain, Proto = prototype selection.

gains 6% and 4% improvement over PMM for popular tags. While SVM is comparable to MMSG for popular tags, our algorithm shows a clear edge over SVM for rare tags, with more than 18% improvement. Since rare tags appear in fewer documents, this result gives credibility to the claim that MMSG works well with very few training instances.

Model Selection for Tag Suggestion

Next we quantitatively show how the model selection reflects the performance of tag suggestion. In our framework, model selection involves the decision of (1) the number of prototypes, (2) the covariance function and (3) the hyper-parameters. Since the hyper-parameters are often associated with the covariance function and can be chosen by optimizing the marginal likelihood of the training data, we then focus on how (1) and (2) affect the performance. A common covariance function used for classification is the squared exponential function (SE) in eq.(2). An alternative function takes the form of neural network (NN):

$$\mathbf{K}(\mathbf{x},\mathbf{x}') = \frac{2}{\pi}\sin^{-1}\left(\frac{2\tilde{\mathbf{x}}^T\Sigma\tilde{\mathbf{x}}'}{\sqrt{(1+2\tilde{\mathbf{x}}^T\Sigma\tilde{\mathbf{x}})(1+2\tilde{\mathbf{x}}'^T\Sigma\tilde{\mathbf{x}}')}}\right), \tag{21}$$

with $\tilde{\mathbf{x}}$ being the augmented vector of the input $\mathbf{x}$.

We compare the results of SVM and our model using these two covariance functions. Figure 9 demonstrates the results on the del.icio.us data (The results on BibSonomy are similar). We set the number of prototypes M to be 5%, 10%, 20% and 50% respectively. It can be observed that MMSG generally outperforms SVM by roughly 10% at each point. With the number of prototypes increases, the precision also soars up from 50% to 62% for MMSG. Meanwhile, by using neural network as the covariance function, both SVM and MMSG gain about 2% precision at each point. Overall, MMSG-NN shows the best performance.

Optimal Prototype Selection for Tag Suggestion

To justify the use of the prototype selection (PS) algorithm, we compare with the criteria used in [17] which efficiently includes points into the active set based on information gain (IG). We also include a random selection (RS) method as the baseline. Figure 10 presents the results on del.icio.us. Generally, prototype selection shows better precision than IG in all four cases. Specifically, prototype selection gains more than 10% performance improvement comparing with information gain when $M = 50\%$.

	Side information for an object
Flickr	*title, description, user_comments, category, additional_information*
Youtube	*title, description, comments, category, name_of_related_videos, videos_from_the_same_person*

Table 6: Side information for training the model.

6.3.2 *Flickr and Youtube*

Finally, we also show the power of MMSG on tagging digital data by using surrounding textual information. We acquired data from Flickr and Youtube between Sep 15 2007 and Oct 21 2007, by subscribing to their RSS feeds of the top 30 most popular tags respectively. We then re-crawled each individual URL in the feeds to get the needed side information listed in Table 6. Overall, the Flickr data contains 22,186 unique items with 10,341 tags whereas Youtube has 2,489 items with 6,724 tags.

For training, the data is organized into 30 classes. Overall, the test results indicate a promising performance. On average, MMSG achieves a 41.6% precision for the two data sets, outperforms PMM and SVM by 5% and 9% respectively. Figure 11 lists several examples with good tagging results by our algorithm. For individual tags, the top-most suggested tags for Flickr have a 56.2% accuracy, compared with a 48.2% accuracy for PMM. Likewise, the top-most tags have 48.2% accuracy for Youtube, 2.3% better than PMM.

Since our algorithm for tag suggestion only leverages textual information, for an image/video without any supporting texts, our algorithm performs no better than a random guess. But since textual information is usually cheap and abundant, our algorithm can serve as a good complement for the content-based approach for digital tagging services.

7. CONCLUSION AND FUTURE WORK

We presented a sparse GP framework for multi-class tag classification. The framework was generalized for multi-label classification for tag suggestion. In particular, we generated real-time tag suggestions for Del.icio.us, BibSonomy,




japan tokyo wood forrest **architecture acient** monks	**birthday** birthdaycake **cakes** candles **people** women **photo**	**flower** tulip tulips red **sky** netherlands **blue** rood	**newyork** skycrapers **usa sky** america **blue buildings glass**
japan architecture tokyo scene building buildings photo **acient**	**birthday cakes people** dinner **photo** party birthparty	**flower** flowers colors color nature **red** rose **blue**	**newyork usa blue** white **buildings glass sky** street



pets animals comedy **fun funny** cute kitten kitty pet	**halo2 halo** mod **game weapon** weapons **xbox** mods	**sports sport tennis roger federer** top 10 best game	**auto vehicle** drift **bmw** m3 track race roadster
animals cats **pets fun** animal **funny** fight music	**game halo2 xbox** live 360 **halo weapon** fun	**sports sport roger tennis** raphi fun match **federer**	**auto** car **vehicle bmw** coupe ford fun movie

Figure 11: Examples of good tags suggested by our algorithm. The first row is the object (image/video). The second row corresponds to the user tags. The third row is the recommended tags by our algorithm.

Flickr and Youtube. Numerous experiments suggested that our method is capable of making good tag suggestion with little training data. Our model successfully outperformed the best tagging algorithm so far [20]. Future work could access how this framework can be extended to semi-supervised tag suggestions and investigate the issue of selecting the optimal number of prototypes for the sparse GP framework. Statistical significance tests will be performed to examine the generalization of our methods to other data sets.

8. REFERENCES

[1] G. Begelman, P. Keller, and F. Smadja. Automated tag clustering: Improving search and exploration in the tag space. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, 2006.

[2] K. Brinker, J. Furnkranz, and E. Hullermeier. A unified model for multilabel classification and ranking. In *ECAI '06*.

[3] P. A. Chirita, S. Costache, W. Nejdl, and S. Handschuh. P-TAG: large scale automatic generation of personalized annotation tags for the web. In *WWW '07*, pages 845–854.

[4] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *NIPS 14*, 2001.

[5] U. Farooq, Y. Song, J. M. Carroll, and C. L. Giles. Social bookmarking for scholarly digital libraries. *IEEE Internet Computing*, pages 29–35, Nov,2007.

[6] M. Girolami and S. Rogers. Variational bayesian multinomial probit regression with gaussian process priors. *Neural Comput.*, 18(8):1790–1817, 2006.

[7] S. Golder and B. Huberman. Usage patterns of collaborative tagging systems. *J. Inf. Sci.*, 2006.

[8] H. Halpin, V. Robu, and H. Shepherd. The complex dynamics of collaborative tagging. In *WWW '07*, 2007.

[9] T. Kohonen. *Self Organization Maps*. Springer, 2001.

[10] M. Kuss and C. E. Rasmussen. Assessing approximate inference for binary gaussian process classification. *J. Mach. Learn. Res.*, 6:1679–1704, 2005.

[11] N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: The informative vector machine. In *NIPS 15*, pages 609–616. 2003.

[12] J. Li and J. Z. Wang. Real-time computerized annotation of pictures. In *MULTIMEDIA '06*, pages 911–920, 2006.

[13] J. Li and H. Zha. Two-way poisson mixture models for simultaneous document classification and word clustering. *Computational Statistics & Data Analysis*, 2006.

[14] J. C. Platt. Probabilities for sv machines. *Advances in Large Margin Classifiers*, pages 61–74, 2000.

[15] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[16] M. Seeger and M. Jordan. Sparse gaussian process classification with multiple classes. TR 661, Department of Statistics, University of California at Berkeley, 2004.

[17] M. Seeger and C. Williams. Fast forward selection to speed up sparse gaussian process regression. In Workshop on AI and Statistics 9, 2003.

[18] S. Seo, M. Bode, and K. Obermayer. Soft nearest prototype classification. *IEEE Trans. Neural Networks*, 2003.

[19] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *NIPS 18*. 2006.

[20] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles. Real-time automatic tag recommendation. In *SIGIR '08*, 2008.

[21] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *Intl. J. of Data Warehousing and Mining*, 3(3):1–13, 2007.

[22] S. Xu, S. Bao, Y. Cao, and Y. Yu. Using social annotations to improve language model for information retrieval. In *CIKM '07*, pages 1003–1006, 2007.

[23] H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *CIKM '01*, 2001.