Preference Learning with Gaussian Processes

Wei Chu CHUWEI@GATSBY.UCL.AC.UK
Zoubin Ghahramani ZOUBIN@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, University College London, London, WC1N 3AR, UK

Abstract

In this paper, we propose a probabilistic kernel approach to preference learning based on Gaussian processes. A new likelihood function is proposed to capture the preference relations in the Bayesian framework. The generalized formulation is also applicable to tackle many multiclass problems. The overall approach has the advantages of Bayesian methods for model selection and probabilistic prediction. Experimental results compared against the constraint classification approach on several benchmark datasets verify the usefulness of this algorithm.

1. Introduction

The statement that x is preferred to y can be simply expressed as an inequality relation $f(x) > f(y)$, where x and y are outcomes or instances, and f defines a preference function. For example, one might prefer "Star Wars" to "Gone with the Wind", or given a news story about the Olympics one might prefer to give it the label "sports" rather than "politics" or "weather". Preference learning has attracted considerable attention in artificial intelligence research (Doyle, 2004). A user's or artificial agent's preferences can guide its decisions and behaviors. In machine learning, the preference learning problem can be restricted to two particular cases: *learning instance preference* and *learning label preference*, as summarized in Fürnkranz and Hüllermeier (2005).

The scenario of learning instance preference consists of a collection of instances $\{x_i\}$ which are associated with a total or partial order relation. Unlike standard supervised learning, the training instances are not assigned a single target; the training data consists of a set of *pairwise preferences between instances*. The goal is to learn the underlying ordering over the instances from these pairwise preferences. Fiechter and Rogers (2000) considered an interesting real-world problem of an adaptive route agent that learns a "subjective" function from preference judgements collected in the traces of user interactions. Bahamonde et al. (2004) studied another challenging problem of evaluating the merits of beef cattle as meat products from the preferences judgements of the experts. The large margin classifiers for preference learning (Herbrich et al., 1998) were widely adapted for the solution. The problem size is the same as the size of pairwise preferences we obtained for training, which is usually *much larger* than the number of distinct instances.

In label preference learning tasks, the preference relations are observed over the predefined set of labels for each instance instead of over instances. Many interesting multiclass problems can be cast in the general framework of label ranking, e.g. multiclass single-label classification, multiclass multilabel classification and hierarchical multiclass categorization etc. Har-Peled et al. (2002) presented a constraint classification approach that provides a general framework for multi-classification and ranking problems based on binary classifiers. Fürnkranz and Hüllermeier (2003) proposed a pairwise ranking algorithm by decomposing the original problem to a set of binary classification problems, one for each pair of labels. Dekel et al. (2004) presented a boosting-based learning algorithm for the label ranking problem using log-linear models. Aiolli and Sperduti (2004) discussed a preference learning model using large margin kernel machines.

Most of the existent algorithms (Aiolli & Sperduti, 2004; Har-Peled et al., 2002; Herbrich et al., 1998) solve the original problem as an augmented binary classification problem. Based on our recent work on ordinal regression (Chu & Ghahramani, 2004), we further develop the Gaussian process algorithm for preference learning tasks. Although the basic techniques we used in these two works are similar, the formulation proposed in this paper is new, more general, and could be applied to tackle many multiclass classification problems. A novel likelihood function is pro-

Appearing in *Proceedings of the 22^{nd} International Conference on Machine Learning*, Bonn, Germany, 2005.

posed for preference learning. The problem size of this approach remains linear with the size of the training instances, rather than growing quadratically. This approach provides a general Bayesian framework for model adaptation and probabilistic prediction. The results of numerical experiments compared against that of the constraint classification approach of Har-Peled et al. (2002) verify the usefulness of our algorithm.

The paper is organized as follows. In section 2 we describe the probabilistic approach for preference learning over instances in detail. In section 3 we generalize this framework to learn label preferences. In section 4, we empirically study the performance of our algorithm on three learning tasks, and we conclude in section 5.

2. Learning Instance Preference

Consider a set of n distinct instances $x_i \in \mathcal{R}^d$ denoted as $\mathcal{X} = \{x_i : i = 1, \ldots, n\}$, and a set of m observed pairwise preference relations on the instances, denoted as

$$\mathcal{D} = \{v_k \succ u_k : k = 1, \ldots, m\} \tag{1}$$

where $v_k \in \mathcal{X}$, $u_k \in \mathcal{X}$, and $v_k \succ u_k$ means the instance v_k is preferred to u_k. For example, the pair (v_k, u_k) could be two options provided by the automated agent for routing (Fiechter & Rogers, 2000), while the user may decide to take the route v_k rather than the route u_k by his/her own judgement.

2.1. Bayesian Framework

The main idea is to assume that there is an unobservable latent function value $f(x_i)$ associated with each training sample x_i, and that the function values $\{f(x_i)\}$ preserve the preference relations observed in the dataset. We impose a Gaussian process prior on these latent function values, and employ an appropriate likelihood function to learn from the pairwise preferences between samples. The Bayesian framework is described with more details in the following.

2.1.1. Prior Probability

The latent function values $\{f(x_i)\}$ are assumed to be a realization of random variables in a zero-mean Gaussian process (Williams & Rasmussen, 1996). The Gaussian processes can then be fully specified by the covariance matrix. The covariance between the latent functions corresponding to the inputs x_i and x_j can be defined by any Mercer kernel functions (Schölkopf & Smola, 2002). A simple example is the Gaussian kernel defined as

$$\mathcal{K}(x_i, x_j) = \exp\left(-\frac{\kappa}{2}\sum_{\ell=1}^{d}(x_i^\ell - x_j^\ell)^2\right) \tag{2}$$

where $\kappa > 0$ and x_i^ℓ denotes the ℓ-th element of x_i. Thus the prior probability of these latent function values $\{f(x_i)\}$ is a multivariate Gaussian

$$\mathcal{P}(\boldsymbol{f}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f}\right) \tag{3}$$

where $\boldsymbol{f} = [f(x_1), f(x_2), \ldots, f(x_n)]^T$, and Σ is the $n \times n$ covariance matrix whose ij-th element is the covariance function $\mathcal{K}(x_i, x_j)$ defined as in (2).

2.1.2. Likelihood

A new likelihood function is proposed to capture the preference relations in (1), which is defined as follows for ideally noise-free cases:

$$\mathcal{P}_{\text{ideal}}\left(v_k \succ u_k | f(v_k), f(u_k)\right) = \begin{cases} 1 & \text{if } f(v_k) \geq f(u_k) \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

This requires that the latent function values of the instances should be consistent with their preference relations. To allow some tolerance to noise in the inputs or the preference relations, we could assume the latent functions are contaminated with Gaussian noise.[1] The Gaussian noise is of zero mean and unknown variance σ^2. $\mathcal{N}(\delta; \mu, \sigma^2)$ is used to denote a Gaussian random variable δ with mean μ and variance σ^2 henceforth. Then the likelihood function (4) becomes

$$\begin{aligned} &\mathcal{P}\left(v_k \succ u_k | f(v_k), f(u_k)\right) \\ &= \int\int \mathcal{P}_{\text{ideal}}(v_k \succ u_k | f(v_k) + \delta_v, f(u_k) + \delta_u) \\ &\qquad \mathcal{N}(\delta_v; 0, \sigma^2)\mathcal{N}(\delta_u; 0, \sigma^2)\, d\delta_v\, d\delta_u \\ &= \Phi\left(z_k\right) \end{aligned} \tag{5}$$

where $z_k = \frac{f(v_k) - f(u_k)}{\sqrt{2}\sigma}$ and $\Phi(z) = \int_{-\infty}^{z}\mathcal{N}(\gamma; 0, 1)d\gamma$. In optimization-based approaches to machine learning, the quantity $-\ln\mathcal{P}\left(v_k \succ u_k | f(v_k), f(u_k)\right)$ is usually referred to as the loss function, i.e. $-\ln\Phi(z_k)$. The derivatives of the loss functions with respect to $f(v_k)$ and $f(u_k)$ are needed in Bayesian methods. The first and second order derivatives of the loss function can be written as

$$\frac{\partial - \ln\Phi(z_k)}{\partial f(x_i)} = \frac{-s_k(x_i)}{\sqrt{2}\sigma}\frac{\mathcal{N}(z_k; 0, 1)}{\Phi(z_k)}, \tag{6}$$

$$\frac{\partial^2 - \ln\Phi(z_k)}{\partial f(x_i)\partial f(x_j)} = \frac{s_k(x_i)s_k(x_j)}{2\sigma^2}\left(\frac{\mathcal{N}^2(z_k; 0, 1)}{\Phi^2(z_k)} + \frac{z_k\mathcal{N}(z_k; 0, 1)}{\Phi(z_k)}\right) \tag{7}$$

where $s_k(x)$ is an indicator function which is $+1$ if $x = v_k$; -1 if $x = u_k$; 0 otherwise.

The likelihood is the joint probability of observing the preference relations given the latent function values, which can be evaluated as a product of the likelihood function (5), i.e.

$$\mathcal{P}(\mathcal{D}|\boldsymbol{f}) = \prod_{k=1}^{m}\mathcal{P}\left(v_k \succ u_k | f(v_k), f(u_k)\right). \tag{8}$$

[1] In principle, any distribution rather than a Gaussian can be assumed for the noise on the latent functions.

2.1.3. Posterior Probability

Based on Bayes' theorem, the posterior probability can then be written as

$$\mathcal{P}(\boldsymbol{f}|\mathcal{D}) = \frac{\mathcal{P}(\boldsymbol{f})}{\mathcal{P}(\mathcal{D})} \prod_{k=1}^{m} \mathcal{P}\left(v_k \succ u_k | f(v_k), f(u_k)\right) \quad (9)$$

where the prior probability $\mathcal{P}(\boldsymbol{f})$ is defined as in (3), the likelihood function is defined as in (5), and the normalization factor $\mathcal{P}(\mathcal{D}) = \int \mathcal{P}(\mathcal{D}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})d\boldsymbol{f}$.

The Bayesian framework we described above is conditional on the model parameters including the kernel parameters κ in the covariance function (2) that control the kernel shape, and the noise level σ in the likelihood function (5). These parameters can be collected into $\boldsymbol{\theta}$, which is the hyperparameter vector. The normalization factor $\mathcal{P}(\mathcal{D})$, more exactly $\mathcal{P}(\mathcal{D}|\boldsymbol{\theta})$, is known as the evidence for the hyperparameters. In the next section, we discuss techniques for hyperparameter learning.

2.2. Model Selection

In a full Bayesian treatment, the hyperparameters $\boldsymbol{\theta}$ must be integrated over the $\boldsymbol{\theta}$-space for prediction. Monte Carlo methods (e.g. Neal, 1996) can be adopted here to approximate the integral effectively. However these might be computationally prohibitive to use in practice. Alternatively, we consider model selection by determining an optimal setting for $\boldsymbol{\theta}$. The optimal values of the hyperparameters can be simply inferred by maximizing the evidence, i.e. $\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \mathcal{P}(\mathcal{D}|\boldsymbol{\theta})$. A popular idea for computing the evidence is to approximate the posterior distribution $\mathcal{P}(\boldsymbol{f}|\mathcal{D})$ as a Gaussian, and then the evidence can be calculated by an explicit formula. In this section, we applied the Laplace approximation (MacKay, 1994) in evidence evaluation. The evidence can be calculated analytically after applying the Laplace approximation at the maximum a posteriori (MAP) estimate, and gradient-based optimization methods can then be employed to implement model adaptation by maximizing the evidence.

2.2.1. maximum a posteriori estimate

The MAP estimate of the latent function values refers to the mode of the posterior distribution, i.e. $\boldsymbol{f}_{\text{MAP}} = \arg\max_{\boldsymbol{f}} \mathcal{P}(\boldsymbol{f}|\mathcal{D})$, which is equivalent to the minimizer of the following functional:

$$\mathcal{S}(\boldsymbol{f}) = -\sum_{k=1}^{m} \ln \Phi(z_k) + \frac{1}{2}\boldsymbol{f}^T \Sigma^{-1} \boldsymbol{f}. \quad (10)$$

Lemma 1. The minimization of the functional $\mathcal{S}(\boldsymbol{f})$, defined as in (10), is a convex programming problem.

Proof. The Hessian matrix of $\mathcal{S}(\boldsymbol{f})$ can be written as $\frac{\partial^2 \mathcal{S}(\boldsymbol{f})}{\partial \boldsymbol{f} \partial \boldsymbol{f}^T} = \Sigma^{-1} + \Lambda$ where Λ is an $n \times n$ matrix whose ij-th entry is $\frac{\partial^2 \sum_{k=1}^m - \ln \Phi(z_k)}{\partial f(x_i) f(x_j)}$. From Mercer's theorem (Schölkopf & Smola, 2002), the covariance matrix Σ is positive semidefinite. The matrix Λ can be shown to be positive semidefinite too as follows. Let $\boldsymbol{y}$ denote a column vector $[y_1, y_2, \ldots, y_n]^T$, and assume the pair (v_k, u_k) in the k-th preference relation is associated with the ν-th and φ-th samples. By exploiting the property of the second order derivative (7), we have $\boldsymbol{y}^T \Lambda \boldsymbol{y} = \sum_{k=1}^m \left(\frac{\partial^2 - \ln \Phi(z_k)}{\partial^2 f(x_\nu)} (y_\nu - y_\varphi)^2 \right)$ and $\frac{\partial^2 - \ln \Phi(z_k)}{\partial^2 f(x_\nu)} > 0 \ \forall f(x_\nu) \in \mathcal{R}$. So $\boldsymbol{y}^T \Lambda \boldsymbol{y} \geq 0$ holds $\forall \boldsymbol{y} \in \mathcal{R}^n$.[2] Therefore the Hessian matrix is a positive semidefinite matrix. This proves the lemma.

The Newton-Raphson formula can be used to find the solution for simple cases. As $\frac{\partial \mathcal{S}(\boldsymbol{f})}{\partial \boldsymbol{f}}|_{\boldsymbol{f}_{\text{MAP}}} = 0$ at the MAP estimate, we have

$$\boldsymbol{f}_{\text{MAP}} = \Sigma \boldsymbol{\beta} \quad (11)$$

where $\boldsymbol{\beta} = \frac{\partial \sum_{k=1}^m \ln \Phi(z_k)}{\partial \boldsymbol{f}}|_{\boldsymbol{f}_{\text{MAP}}}$.

2.2.2. Evidence Approximation

The Laplace approximation of $\mathcal{S}(\boldsymbol{f})$ refers to carrying out the Taylor expansion at the MAP point and retaining the terms up to the second order (MacKay, 1994). This is equivalent to approximating the posterior distribution $\mathcal{P}(\boldsymbol{f}|\mathcal{D})$ as a Gaussian distribution centered on $\boldsymbol{f}_{\text{MAP}}$ with the covariance matrix $(\Sigma^{-1} + \Lambda_{\text{MAP}})^{-1}$, where Λ_{MAP} denotes the matrix Λ at the MAP estimate. The evidence can then be computed as an explicit expression, i.e.

$$\mathcal{P}(\mathcal{D}|\boldsymbol{\theta}) \approx \exp(-\mathcal{S}(\boldsymbol{f}_{\text{MAP}}))|\mathbf{I} + \Sigma\Lambda_{\text{MAP}}|^{-\frac{1}{2}} \quad (12)$$

where $\mathbf{I}$ is an $n \times n$ identity matrix. The quantity (12) is a convenient yardstick for model selection.

2.2.3. Gradient Descent

Grid search can be used to find the optimal hyperparameter values $\boldsymbol{\theta}^\star$, but such an approach is very expensive when a large number of hyperparameters are involved. For example, automatic relevance determination (ARD) parameters [3] could be embedded into the covariance function (2) as a means of feature selection. The ARD Gaussian kernel can be defined as

$$\mathcal{K}(x_i, x_j) = \exp\left(-\frac{1}{2}\sum_{\ell=1}^d \kappa^\ell (x_i^\ell - x_j^\ell)^2\right) \quad (13)$$

[2] Practically we can insert a "jitter" term on the diagonal entries of the matrix to make it positive definite.

[3] The techniques of *automatic relevance determination* were originally proposed by MacKay (1994) and Neal (1996) in the context of Bayesian neural networks as a hierarchical prior over the weights.

where $\kappa^\ell > 0$ is the ARD parameter for the ℓ-th feature that controls the contribution of this feature in the modelling. The number of hyperparameters increases to $d+1$ in this case.

Gradient-based optimization methods are regarded as suitable tools to determine the values of these hyperparameters, as the gradients of the logarithm of the evidence (12) with respect to the hyperparameters $\boldsymbol{\theta}$ can be derived analytically. We usually collect $\{\ln\sigma, \ln\kappa\}$ as the set of variables to tune. This definition of tunable variables is helpful to convert the constrained optimization problem into an unconstrained optimization problem. The gradients of $\ln\mathcal{P}(\mathcal{D}|\boldsymbol{\theta})$ with respect to these variables can be derived as follows:

$$\begin{aligned}\frac{\partial\ln\mathcal{P}(\mathcal{D}|\boldsymbol{\theta})}{\partial\ln\kappa} &= \frac{\kappa}{2}\boldsymbol{f}_{\mathrm{MAP}}^T\Sigma^{-1}\frac{\partial\Sigma}{\partial\kappa}\Sigma^{-1}\boldsymbol{f}_{\mathrm{MAP}} \\ &-\frac{\kappa}{2}\mathrm{trace}\left[(\Sigma^{-1}+\Lambda_{\mathrm{MAP}})^{-1}\Sigma^{-1}\frac{\partial\Sigma}{\partial\kappa}\Lambda_{\mathrm{MAP}}\right] \\ &-\frac{\kappa}{2}\mathrm{trace}\left[(\Sigma^{-1}+\Lambda_{\mathrm{MAP}})^{-1}\frac{\partial\Lambda_{\mathrm{MAP}}}{\partial\kappa}\right],\end{aligned} \tag{14}$$

$$\begin{aligned}\frac{\partial\ln\mathcal{P}(\mathcal{D}|\boldsymbol{\theta})}{\partial\ln\sigma} &= \sigma\sum_{k=1}^{m}\frac{\partial\ln\Phi\left(\frac{f_{\mathrm{MAP}}(v_k)-f_{\mathrm{MAP}}(u_k)}{\sqrt{2}\sigma}\right)}{\partial\sigma} \\ &-\frac{\sigma}{2}\mathrm{trace}\left[(\Sigma^{-1}+\Lambda_{\mathrm{MAP}})^{-1}\frac{\partial\Lambda_{\mathrm{MAP}}}{\partial\sigma}\right].\end{aligned} \tag{15}$$

Then gradient-descent methods can be employed to search for the maximizer of the log evidence.

2.3. Prediction

Now let us take a test pair (r,s) on which the preference relation is unknown. The zero-mean latent variables $\boldsymbol{f}_t = [f(r), f(s)]^T$ have correlations with the n zero-mean random variables of training samples $\{f(x_i)\}_{i=1}^n$.[4] The correlations are defined by the covariance function in (2), so that we have the prior joint multivariate Gaussian distribution, i.e.

$$\begin{bmatrix}\boldsymbol{f}\\ \boldsymbol{f}_t\end{bmatrix} \sim \mathcal{N}\left[\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}\Sigma & \boldsymbol{k}_t\\ \boldsymbol{k}_t^T & \Sigma_t\end{pmatrix}\right]$$

where $\boldsymbol{k}_t = \begin{bmatrix}\mathcal{K}(r,x_1),\mathcal{K}(r,x_2),\ldots,\mathcal{K}(r,x_n)\\ \mathcal{K}(s,x_1),\mathcal{K}(s,x_2),\ldots,\mathcal{K}(s,x_n)\end{bmatrix}^T$ and $\Sigma_t = \begin{bmatrix}\mathcal{K}(r,r) & \mathcal{K}(r,s)\\ \mathcal{K}(s,r) & \mathcal{K}(s,s)\end{bmatrix}$. So the conditional distribution $\mathcal{P}(\boldsymbol{f}_t|\boldsymbol{f})$ is a Gaussian too. The predictive distribution of $\mathcal{P}(\boldsymbol{f}_t|\mathcal{D})$ can be computed as an integral over $\boldsymbol{f}$-space, which can be written as

$$\mathcal{P}(\boldsymbol{f}_t|\mathcal{D}) = \int\mathcal{P}(\boldsymbol{f}_t|\boldsymbol{f})\mathcal{P}(\boldsymbol{f}\,|\mathcal{D})\,d\boldsymbol{f}. \tag{16}$$

The posterior distribution $\mathcal{P}(\boldsymbol{f}|\mathcal{D})$ can be approximated as a Gaussian by the Laplace approximation. The predictive distribution (16) can be finally simplified as a Gaussian $\mathcal{N}(\boldsymbol{f}_t;\mu^\star,\Sigma^\star)$ with mean

$$\mu^\star = [\mu_r^\star, \mu_s^\star]^T = \boldsymbol{k}_t^T\Sigma^{-1}\boldsymbol{f}_{\mathrm{MAP}} = \boldsymbol{k}_t^T\boldsymbol{\beta} \tag{17}$$

and variance

$$\Sigma^\star = \begin{bmatrix}\Sigma_{rr}^\star & \Sigma_{rs}^\star\\ \Sigma_{sr}^\star & \Sigma_{ss}^\star\end{bmatrix} = \Sigma_t - \boldsymbol{k}_t^T(\mathbf{I}+\Lambda_{\mathrm{MAP}}\Sigma)^{-1}\Lambda_{\mathrm{MAP}}\boldsymbol{k}_t. \tag{18}$$

The predictive preference $\mathcal{P}(r \succ s|\mathcal{D})$ can be evaluated by the integral $\int\mathcal{P}(r\succ s|\boldsymbol{f}_t,\mathcal{D})\mathcal{P}(\boldsymbol{f}_t|\mathcal{D})d\boldsymbol{f}_t$ that yields

$$\mathcal{P}(r\succ s|\mathcal{D}) = \Phi\left(\frac{\mu_r^\star-\mu_s^\star}{\sigma_\star}\right) \tag{19}$$

where $\sigma_\star^2 = 2\sigma^2 + \Sigma_{rr}^\star + \Sigma_{ss}^\star - \Sigma_{rs}^\star - \Sigma_{sr}^\star$.

2.4. Discussion

The evidence evaluation (12) involves solving a convex programming problem (10) and then computing the determinant of an $n\times n$ matrix, which costs CPU time at $\mathcal{O}(n^3)$, where n is the number of distinct instances in the preference pairs for training which is potentially much fewer than the number of preference relations m. Active learning can be applied to learn on very large datasets efficiently (Brinker, 2004). The fast training algorithm for Gaussian processes (Csató & Opper, 2002) can also be adapted in the settings of preference learning for speedup. Lawrence et al. (2002) proposed a greedy selection criterion rooted in information-theoretic principles for sparse representation of Gaussian processes. In the Bayesian framework we have described, the expected informativeness of a new pairwise preference relation can be measured as the change in entropy of the posterior distribution of the latent functions by the inclusion of this preference. A promising approach to active learning is to select from the data pool the sample with the highest expected information gain. This is a direction for future work.

3. Learning Label Preference

Preference relations can be defined over the instances' labels instead of over the instances. In this case, each instance is associated with a predefined set of labels, and the preference relations over the label set are defined. This learning task is also known as *label ranking*. The preferences of each training sample can be presented in the form of a directed graph, known as a preference graph (Dekel et al., 2004; Aiolli & Sperduti, 2004), where the labels are the graph vertices. The preference graph can be decomposed into a set of pairwise preference relations over the label set for each sample. In Figure 1, we present three popular examples as an illustration.

Suppose that we are provided with a training dataset $\{x_i, \mathcal{E}_i\}_{i=1}^n$. $x_i \in \mathcal{R}^d$ is a sample for training and $\mathcal{E}_i$ is the set of directed edges in the preference graph

[4]The latent variables $f(r)$ and $f(s)$ are assumed to be distinct from $\{f(x_i)\}_{i=1}^n$.

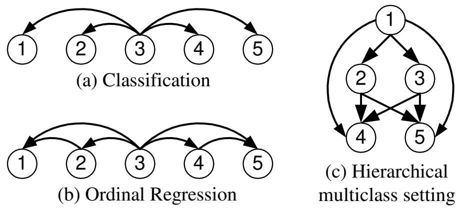

Figure 1. Graphs of label preferences, where an edge from node i to node j indicates that label i is preferred to label j. (a) standard multiclass classification where 3 is the correct label. (b) the case of ordinal regression where 3 is the correct ordinal scale. (c) a multi-layer graph in hierarchical multiclass settings that specifies three levels of label preferences.

for x_i, denoted as $\mathcal{E}_i = \{c_i^{j+} \leftarrow c_i^{j-}\}_{j=1}^{g_i}$ where c_i^{j-} is the initial label vertex of the j-th edge while c_i^{j+} is the terminal label, and g_i is the number of edges. Each sample can have a different preference graph over the labels. The Bayesian framework for instance preferences can be generalized to learn label preferences similarly to Gaussian processes for multiclass classification (Williams & Barber, 1998). We introduce distinct Gaussian processes for each predefined label, and the label preference of the samples are preserved by the latent function values in these Gaussian processes via the likelihood function (5).

The prior probability of these latent functions $\mathcal{P}(\boldsymbol{f})$ becomes a product of multivariate Gaussians, i.e.

$$\prod_{a=1}^{L} \mathcal{P}(\boldsymbol{f}_a) = \prod_{a=1}^{L} \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_a|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\boldsymbol{f}_a^T \Sigma_a^{-1} \boldsymbol{f}_a\right) \tag{20}$$

where $\boldsymbol{f}_a = [f_a(x_1), f_a(x_2), \ldots, f_a(x_n)]$ and L is the number of the labels. Σ_a is the covariance matrix defined by the kernel function as in (2). The observed edges $\{c_i^{j+} \leftarrow c_i^{j-}\}_{j=1}^{g_i}$ require the corresponding function values $\{f_{c_i^{j+}}(x_i) \geq f_{c_i^{j-}}(x_i)\}_{j=1}^{g_i}$. Using the likelihood function for pairwise preferences (5), the likelihood of observing these preference graphs can be computed as

$$\mathcal{P}(\boldsymbol{\mathcal{E}}|\boldsymbol{f}) = \prod_{i=1}^{n}\prod_{j=1}^{g_i} \Phi(z_i^j) \tag{21}$$

where $z_i^j = \frac{1}{\sqrt{2}\sigma}\left(f_{c_i^{j+}}(x_i) - f_{c_i^{j-}}(x_i)\right)$ and $\Phi(\pi) = \int_{-\infty}^{\pi} \mathcal{N}(\gamma; 0, 1) d\gamma$. The posterior probability can then be written as

$$\mathcal{P}(\boldsymbol{f}|\boldsymbol{\mathcal{E}}) = \frac{1}{\mathcal{P}(\boldsymbol{\mathcal{E}})} \prod_{i=1}^{n}\prod_{j=1}^{g_i} \Phi(z_i^j) \prod_{a=1}^{L} \mathcal{P}(\boldsymbol{f}_a) \tag{22}$$

where $\mathcal{P}(\boldsymbol{\mathcal{E}}) = \int \mathcal{P}(\boldsymbol{\mathcal{E}}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f}) d\boldsymbol{f}$ is the model evidence.

Approximate Bayesian methods can be applied to infer the optimal hyperparameter $\boldsymbol{\theta}$. We applied the Laplace approximation again to evaluate the evidence. The MAP estimate is equivalent to the solution to the following optimization problem:

$$\min_{\boldsymbol{f}} \check{\mathcal{S}}(\boldsymbol{f}) = \frac{1}{2}\sum_{a=1}^{L} \boldsymbol{f}_a^T \Sigma_a^{-1} \boldsymbol{f}_a - \sum_{i=1}^{n}\sum_{j=1}^{g_i} \ln \Phi(z_i^j) \tag{23}$$

Like (10), this is also a convex programming problem. At the MAP estimate we have

$$\boldsymbol{f}_a^{\mathrm{MAP}} = \Sigma_a \boldsymbol{\beta}_a \tag{24}$$

where $\boldsymbol{\beta}_a = \frac{\partial \sum_{i=1}^{n}\sum_{j=1}^{g_i} \ln \Phi(z_i^j)}{\partial \boldsymbol{f}_a}|_{\boldsymbol{f}_a^{\mathrm{MAP}}}$. The evidence $\mathcal{P}(\boldsymbol{\mathcal{E}})$, more exactly $\mathcal{P}(\boldsymbol{\mathcal{E}}|\boldsymbol{\theta})$, with the Laplace approximation (MacKay, 1994), can be approximated as the following expression accordingly:

$$\mathcal{P}(\boldsymbol{\mathcal{E}}|\boldsymbol{\theta}) \approx \exp(-\check{\mathcal{S}}(\boldsymbol{f}_{\mathrm{MAP}}))|\mathbf{I} + \check{\Sigma}\check{\Lambda}_{\mathrm{MAP}}|^{-\frac{1}{2}} \tag{25}$$

where $\mathbf{I}$ is an $nL \times nL$ identity matrix, $\check{\Lambda}_{\mathrm{MAP}} = \frac{\partial^2 - \sum_{i=1}^{n}\sum_{j=1}^{g_i} \ln \Phi(z_i^j)}{\partial \boldsymbol{f} \partial \boldsymbol{f}^T}|_{\boldsymbol{f}_{\mathrm{MAP}}}$ and $\check{\Sigma}$ is an $nL \times nL$ block-diagonal matrix with blocks $\{\Sigma_a\}$. The gradients with respect to $\boldsymbol{\theta}$ can be derived as in (14)– (15) accordingly. The optimal hyperparameters $\boldsymbol{\theta}^\star$ can be discovered by a gradient-descent optimization package.

During prediction, the test case x_t is associated with L latent functions $\{f_a(x_t)\}_{a=1}^{L}$ for the predefined labels respectively. The correlations between $f_a(x_t)$ and $\boldsymbol{f}_a$ are defined by the kernel function as in (2), i.e. $\check{\boldsymbol{k}}_t = [\mathcal{K}_a(x_t, x_1), \mathcal{K}_a(x_t, x_2), \ldots, \mathcal{K}_a(x_t, x_n)]$.[5] The mean of the predictive distribution $\mathcal{P}(f_a(x_t)|\boldsymbol{\mathcal{E}}, \theta^\star)$ can be approximated as $E[f_a(x_t)] = \check{\boldsymbol{k}}_t \boldsymbol{\beta}_a$ where $\boldsymbol{\beta}_a$ is defined as in (24) at $\boldsymbol{\theta}^\star$. The label preference can then be decided by

$$\arg\mathrm{sort}_{a=1,\ldots,L} E[f_a(x_t)]. \tag{26}$$

This label preference learning method can be applied to tackle ordinal regression using the preference graph in Figure 1(b). Such an approach is different from our previous work on ordinal regression (Chu & Ghahramani, 2004). Both methods use Gaussian process prior and implement ordering information by inequalities. However the above approach needs L Gaussian processes while the approach in Chu and Ghahramani (2004) uses only a single Gaussian process. For ordinal regression problems, the approach in Chu and Ghahramani (2004) seems more natural.

4. Numerical Experiments

In the implementation of our Gaussian process algorithm for preference learning, gradient-descent meth-

[5] In the current work, we simply constrained all the covariance functions to use the same kernel parameters.

ods have been employed to maximize the approximated evidence for model adaptation.[6] We started from the initial values of the hyperparameters to infer the optimal ones.[7] We also implemented the constraint classification method of Har-Peled et al. (2002) using support vector machines for comparison purpose (CC-SVM). 5-fold cross validation was used to determine the optimal values of model parameters (the Gaussian kernel κ and the regularization factor) involved in the CC-SVM formulation, and the test error was obtained using the optimal model parameters for each formulation. The initial search was done on a 7×7 coarse grid linearly spaced by 1.0 in the region $\{(\log_{10} C, \log_{10} \kappa)| - 2 \le \log_{10} C \le 4, -3 \le \log_{10} \kappa \le 3\}$, followed by a fine search on a 9×9 uniform grid linearly spaced by 0.2 in the $(\log_{10} C, \log_{10} \kappa)$ space. We begin this section to compare the generalization performance of our algorithm against the CC-SVM approach on five datasets of instance preferences. Then we empirically study the scaling properties of the two algorithms on an information retrieval data set. We also apply our algorithm to several classification and label ranking tasks to verify the usefulness.

4.1. Instance Preference

We first compared the performance of our algorithm against the CC-SVM approach on the tasks of learning instance preferences. We collected five benchmark datasets that were used for metric regression problems.[8] The target values were used to decide the preference relations between pairs of instances. For each dataset, we randomly selected a number of training pairs as specified in Table 1, and 20000 pairs for testing. The selection was repeated 20 times independently. The Gaussian kernel (2) was used for both the CC-SVM and our algorithm. In the CC-SVM algorithm, each preference relation $x_i \succ x_j$ is transformed to a pair of new samples with labels $+1$ and -1 respectively. We report their test results in Table 1, along with the results of our algorithm using the ARD Gaussian kernel (13). The GP algorithm gives significantly better test results than that of the CC-SVM approach on three of the five datasets. The ARD kernel yields better performance on the Boston Housing and comparable results on other datasets.

Table 1. Test results on the five datasets for preference learning. "Error Rate" is the percent of incorrect preference prediction averaged over 20 trials along with standard deviation. "m" is the number of training pairs and "d" is the input dimension. "CC-SVM" and "GP" denotes the CC-SVM and our algorithm using the Gaussian kernel. "GPARD" denotes our algorithm using the ARD Gaussian kernel. We use bold face to indicate the lowest error rate. The symbols $\star$ indicate the cases of CC-SVM significantly worse than that of GP; A p-value threshold of 0.01 in Wilcoxon rank sum test was used to decide this.

			Error Rate (%)		
Dataset	m	d	CC-SVM	GP	GPARD
Pyrimidines	100	27	16.01±2.29	**14.43±2.02**	16.56±1.76
Triazines	300	60	20.37±1.32*	**17.78±0.97**	18.26±1.45
MachineCpu	500	6	15.31±1.26*	**12.12±1.49**	12.86±0.71
BostonHouse	700	13	13.30±1.08	12.85±0.46	**10.44±0.64**
Abalone	1000	10	18.76±0.35*	**17.29±0.38**	17.35±0.38

Hersh et al. (1994) generated the OHSUMED dataset for information retrieval, where the relevance level of the documents with respect to the given textual query were assessed by human experts, using three rank scales: definitely relevant, possibly relevant or not relevant. In our experiment to study the scaling properties, we used the results of "Query 3" in OHSUMED that contain 201 references taken from the whole database (99 definitely, 59 possibly, and 43 irrelevant). The bag-of-words representation was used to translate these documents into the vectors of "term frequency"(TF) components scaled by "inverse document frequencies"(IDF). We used the "Rainbow" software released by McCallum (1996) to scan the title and abstract of these documents to compute the TFIDF vectors. In the preprocessing, we skipped the terms in the "stoplist",[9] and restricted ourselves to terms that appear in at least 3 of the 201 documents. So each document is represented by its TFIDF vector with 576 distinct elements. To account for different document lengths, we normalized the length of each document vector to unity. The preference relation of a pair of documents can be determined by their rank scales. We randomly selected a subset of pairs of documents (having different rank scales) with size $\{100, 200, \ldots, 1000\}$ for training, and then tested on the remaining pairs. At each size, the random selection was carried out 20 times. The linear kernel $\mathcal{K}(x_i, x_j) = \sum_{\ell=1}^{d} x_i^\ell x_j^\ell$ was used for the CC-SVM and our algorithm. The test results of the two algorithms are presented in the left graph of Figure 2. The performances of the two algorithms are very competitive on this application. In the right graph of Figure 2, the circles present the CPU time consumed to solve (10) and evaluate the evidence (12) once in our algorithm

[6] The source code written in ANSI C can be found at http://www.gatsby.ucl.ac.uk/~chuwei/plgp.htm.

[7] In numerical experiments, the initial values of the hyperparameters were usually chosen as $\sigma = 1.0$ and $\kappa = 1/d$ where d is the input dimension. We suggest to try more starting points in practice and then choose the best model by the evidence.

[8] These regression datasets are available at http://www.liacc.up.pt/~ltorgo/Regression/DataSets.html.

[9] The "stoplist" is the SMART systems' list of 524 common words, like "the" and "of".

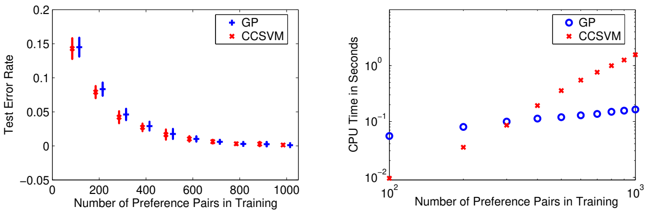

Figure 2. The left graph presents the test error rates on preference relations of the OHSUMED dataset at different training data sizes. The crosses indicate the average values over the 20 trials and the vertical lines indicate the standard deviation. The right graph presents the CPU time in seconds consumed by the two algorithms.

while the crosses present the CPU time for solving the quadratic programming problem once in the CC-SVM approach. We observed that the computational cost of the CC-SVM approach is dependent on the number of preference pairs in training with scaling exponent about 2.2, whereas the overhead of our algorithm is almost independent of the number of training preference pairs. As we have discussed in Section 2.4, the complexity of our algorithm is mainly dependent on the number of distinct instances involved in the training data. Since the number of pairwise preferences for training is usually *much larger* than the number of instances, the computational advantage is one of the merits of our algorithm over the CC-SVM-like algorithms.

4.2. Classification

Next, we selected five benchmark datasets for multiclass classification used by Wu et al. (2004) and applied both the CC-SVM and our algorithm on these tasks. All the datasets contain 300 training samples and 500 test samples. The partitions were repeated 20 times for each dataset.[10] The number of classes and features of the five datasets are recorded in Table 2 denoted by L and d respectively. In our algorithm, the preference graph of each training sample contains $L-1$ edges as depicted in Figure 1(a), and the predictive class can be determined by $\arg\max_a E[f_a(x_t)]$ where $E[f_a(x_t)]$ is defined as in (26). In the CC-SVM algorithm, each training sample is transformed to $2(L-1)$ new samples that represent the $L-1$ pairwise preferences (Har-Peled et al., 2002). The Gaussian kernel (2) was used for both the CC-SVM and our algorithm. We report the test results in Table 2, along with the results of SVM with pairwise coupling cited from the Table 2 of Wu et al. (2004). Our GP approach are very competitive with pairwise coupling SVM and the CC-SVM algorithm on class label prediction, and significantly better than the CC-SVM algorithm in preference prediction on two of the five datasets.

[10]These classification datasets are maintained at www.csie.ntu.edu.tw/∼cjlin/papers/svmprob/data.

Table 2. Test results on the five datasets for standard multiclass classification. "L" is the number of classes and "d" denotes the number of input features. "Label Error Rate" denotes the percent of incorrect predictions on class labels averaged over 20 trials. "Pref Error Rate" denotes the percent of incorrect predictions on preference relations averaged over 20 trials along with standard deviation. "PW" denotes the results of SVM with pairwise coupling cited from Wu et al. (2004). "CC-SVM" and "GP" denotes the CC-SVM and our algorithm using the Gaussian kernel. We use bold face to indicate the lowest error rate. The symbols $\star$ indicate the cases of CC-SVM significantly worse than that of GP; A p-value threshold of 0.01 in Wilcoxon rank sum test was used to decide the statistical significance.

		Label Error Rate (%)			Pref Error Rate (%)	
Dataset	L/d	PW	CC-SVM	GP	CC-SVM	GP
DNA	3/180	**10.47**	10.85	10.67	6.23±0.99	**6.08±0.92**
Waveform	3/21	16.23	16.76	**15.22**	8.39±0.84*	**7.62±0.79**
Satimage	6/36	**14.12**	14.23	15.21	4.84±0.70	**4.03±0.45**
Segment	7/19	**6.21**	6.98	**6.21**	1.66±0.50	**1.41±0.38**
USPS	10/256	**11.57**	13.30	12.13	3.20±0.39*	**2.82±0.39**

4.3. Label Ranking

To test on the label ranking tasks, we used the decision-theoretic settings related to expected utility theory described by Fürnkranz and Hüllermeier (2003). An agent attempts to take one action from a set of alternative actions $A = \{a_1, a_2, \ldots, a_L\}$ with the purpose of maximizing the expected utility under the uncertainty of the world states $W = \{w_1, w_2, \ldots, w_d\}$. The expected utility of act a_i is given by $E(a_i) = \sum_{j=1}^{d} p_j U_{ij}$ where the probability of state w_j is p_j and the utility of acting a_i in the state w_j is $U_{ij} \in [0, 1]$. In our experiment, the set of samples corresponding to the set of probability vectors p, were randomly generated according to a uniform distribution over $\{p \in \mathcal{R}^d | p \geq 0, p_1 + \ldots + p_d = 1\}$. We fixed the number of world states/features $d = 10$ and the number of samples $n = 50$, but varied the number of actions/labels L from 2 to 10. The utility matrix was generated at random by drawing independently and uniformly distributed entries $U_{ij} \in [0, 1]$. At each label size, we independently repeated this procedure 20 times. The two algorithms employed the linear kernel to learn the underlying utility matrix. In our algorithm, the preference graph of each training sample contains $L(L-1)/2$ edges and the label preference for test samples was decided by (26). In the CC-SVM algorithm, each training sample was transformed to $L(L-1)$ new samples with dL augmented features. The preference test rates and averaged Spearman rank correlations are presented in Figure 3. The rank correlation coefficient for each test

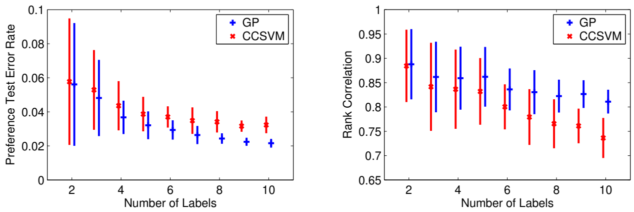

Figure 3. The left graph presents the preference test rates of the two algorithms on the label ranking tasks with different number of labels, while the right presents the averaged rank correlation coefficients. The middle crosses indicate the average values over the 20 trials and the vertical lines indicate the standard deviations.

case is defined as $1 - \frac{6\sum_{a=1}^{d}(l_a - \tilde{l}_a)^2}{d(d^2-1)}$ where l_a is the true rank and $\tilde{l}_a$ is the predictive rank. On this application, our GP algorithm is clearly superior to the CC-SVM approach on generalization capacity, especially when the number of labels becomes large. The potential reason for this observation might be that learning with CC-SVM in the dL-dimensional augmented input space becomes much harder.

5. Conclusions

In this paper we proposed a nonparametric Bayesian approach to preference learning over instances or labels. The formulation of learning label preference is also applicable to many multiclass learning tasks. In both formulations, the problem size remains linear with the number of distinct samples in the training preference pairs. The existent fast algorithms for Gaussian processes can be adapted to tackle large datasets. Experimental results on benchmark datasets show the generalization performance of our algorithm is competitive and often better than the constraint classification approach with support vector machines.

Acknowledgments

This work was supported by the National Institutes of Health and its National Institute of General Medical Sciences division under Grant Number 1 P01 GM63208.

References

Aiolli, F., & Sperduti, A. (2004). Learning preferences for multiclass problems. *Advances in Neural Information Processing Systems 17*.

Bahamonde, A., Bayón, G. F., Díez, J., Quevedo, J. R., Luaces, O., del Coz, J. J., Alonso, J., & Goyache, F. (2004). Feature subset selection for learning preferences: A case study. *Proceedings of the 21th International Conference on Machine Learning* (pp. 49–56).

Brinker, K. (2004). Active learning of label ranking functions. *Proceedings of the 21th International Conference on Machine Learning* (pp. 129–136).

Chu, W., & Ghahramani, Z. (2004). *Gaussian processes for ordinal regression* (Technical Report). Gatsby Computational Neuroscience Unit, University College London. http://www.gatsby.ucl.ac.uk/∼chuwei/paper/gpor.pdf.

Csató, L., & Opper, M. (2002). Sparse online Gaussian processes. *Neural Computation*, *14*, 641–668.

Dekel, O., Keshet, J., & Singer, Y. (2004). Log-linear models for label ranking. *Proceedings of the 21st International Conference on Machine Learning* (pp. 209–216).

Doyle, D. (2004). Prospects of preferences. *Computational Intelligence*, *20*, 111–136.

Fiechter, C.-N., & Rogers, S. (2000). Learning subjective functions with large margins. *Proc. 17th International Conf. on Machine Learning* (pp. 287–294).

Fürnkranz, J., & Hüllermeier, E. (2003). Pairwise preference learning and ranking. *Proceedings of the 14th European Conference on Machine Learning* (pp. 145–156).

Fürnkranz, J., & Hüllermeier, E. (2005). Preference learning. *Künstliche Intelligenz*. in press.

Har-Peled, S., Roth, D., & Zimak, D. (2002). Constraint classification: A new approach to multiclass classification and ranking. *Advances in Neural Information Processing Systems 15*.

Herbrich, R., Graepel, T., Bollmann-Sdorra, P., & Obermayer, K. (1998). Learning preference relations for information retrieval. *Proc. of Workshop Text Categorization and Machine Learning, ICML* (pp. 80–84).

Hersh, W., Buckley, C., Leone, T., & Hickam, D. (1994). Ohsumed: An interactive retrieval evaluation and new large test collection for research. *Proceedings of the 17th Annual ACM SIGIR Conference* (pp. 192–201).

Lawrence, N. D., Seeger, M., & Herbrich., R. (2002). Fast sparse Gaussian process methods: The informative vector machine. *Advances in Neural Information Processing Systems 15* (pp. 609–616).

MacKay, D. J. C. (1994). Bayesian methods for backpropagation networks. *Models of Neural Networks III*, 211–254.

McCallum, A. K. (1996). Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/∼mccallum/bow.

Neal, R. M. (1996). *Bayesian learning for neural networks*. Lecture Notes in Statistics. Springer.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge, MA: The MIT Press.

Williams, C. K. I., & Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *20*, 1342–1351.

Williams, C. K. I., & Rasmussen, C. E. (1996). Gaussian processes for regression. *Advances in Neural Information Processing Systems* (pp. 598–604). MIT Press.

Wu, T.-F., Lin, C.-J., & Weng, R. C. (2004). Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, *5*, 975–1005.