Fast Active Exploration for Link-Based Preference Learning using Gaussian Processes

Zhao Xu[1], Kristian Kersting[1], and Thorsten Joachims[2]

[1] Fraunhofer IAIS, Schloss Birlinghoven, 53754 Sankt Augustin, Germany
{zhao.xu,kristian.kersting}@iais.fraunhofer.de
[2] Department of Computer Science, Cornell University, Ithaca, NY 14853, USA
tj@cs.cornell.edu

Abstract. In preference learning, the algorithm observes pairwise relative judgments (preference) between items as training data for learning an ordering of all items. This is an important learning problem for applications where absolute feedback is difficult to elicit, but pairwise judgments are readily available (e.g., via implicit feedback [13]). While it was already shown that active learning can effectively reduce the number of training pairs needed, the most successful existing algorithms cannot generalize over items or queries. Considering web search as an example, they would need to learn a separate relevance score for each document-query pair from scratch. To overcome this inefficiency, we propose a link-based active preference learning method based on Gaussian Processes (GPs) that incorporates dependency information from both feature-vector representations as well as relations. Specifically, to meet the requirement on computational efficiency of active exploration, we introduce a novel incremental update method that scales as well as the non-generalizing models. The proposed algorithm is evaluated on datasets for information retrieval, showing that it learns substantially faster than algorithms that cannot model dependencies.

1 Introduction

Preference learning is a natural and widely successful problem formulation for applications in search engines, information retrieval, and collaborative filtering [7, 3, 12, 8]. The learning algorithm receives pairwise preferences that compare two entities, such as documents, webpages, products, songs etc. The goal is to learn a general ordering function that also ranks unobserved pairs correctly. Since collecting preference pairs is expensive (i.e. manual judgment effort, or presentation of inferior rankings for implicit feedback), designing learners that *actively* collect the most *informative* training preferences promises to reduce training cost. While some approaches for active learning with preferences exist(e.g., [21, 6, 19, 27, 26]), most assume the entities to be independent of each other and they do not take into account relations between the entities [1]. However, such link structures among entities are very informative [24, 9, 10, 17]. For example, assume that there are n papers related to a query q, and these papers are linked

together into a network with co-author relations. Generally, the papers written by the same authors represent some similarity on research topics. If a user gives a judgement that a paper e_i is more relevant to the query q than another paper e_j ($e_i \succ e_j$), then a co-authored paper $e_{i'}$ of e_i is more likely to have similar relevance. In particular, it is likely to be preferred to e_j ($e_{i'} \succ e_j$) as well. More generally, known preference information on papers propagates through the network providing useful evidence about unknown preferences and in turn decreasing our uncertainty about them. For active learning, this means that the search space is reduced.

Most of the active preference learning methods to date, however, have remained relatively agnostic to this rich structure. In fact, the most successful existing algorithms cannot model dependencies. For instance, Saar-Tsechansky and Provost [21] considered class-based ranking problems and proposed to use bootstrap samples of existing training data to examine the variance in the probability estimates for not-yet-labeled data. Brinker [2] proposed an SVM-based method which converts preference learning into a binary classification problem. [5, 25] also ignore the link structure. In the model proposed by Radlinski and Joachims [19], the lack of dependencies manifests itself in a diagonal covariance matrix. Each entity is associated with a latent variable representing its utility (score) and all latent variables are independent of each other and follow Gaussian distributions. The entities are ranked according to these utilities: an entity e_i is ranked above another one e_j ($e_i \succ e_j$), if and only if the utility of e_i is larger than that of e_j. For active exploration, Radlinski and Joachims propose to select an entity pair for which a preference label promises the largest expected reduction of uncertainty about the latent utilities. Specifically, for a user-defined loss function, the selection criterion optimizes the expected reduction of loss due to the variability of the utility estimates.

The method proposed in this paper overcomes the key limiting assumption of [19], namely that all document-query utilities have zero covariance. Specifically, we propose a link-based active preference learning method using Gaussian processes. It is based on the relational Gaussian process model for preference learning, called XPGP, recently introduced by Kersting and Xu [14]. Like in Radlinski and Joachims' model, associated with each entity is a continuous latent variable ξ_i that represents the latent utility (score) of the entity. The key difference from Radlinski and Joachims' model is that the latent utility (score) of each entity consists of two function values: $f(x_i)$ and $g(r_i)$. With each entity e_i described by a vector of entity attributes x_i, $f(x_i)$ is a function value of these attributes. The $\{f(x_1), f(x_2), \ldots\}$ share a common Gaussian process prior GP_a. The term $g(r_i)$ denotes a function value of entity relations r_i between e_i and other entities, and $\{g(r_1), g(r_2), \ldots\}$ share another common Gaussian process prior GP_r. The utility ξ_i of e_i is then modeled as the weighted sum of the two components: $\xi_i = \omega_1 f(x_i) + \omega_2 g(r_i)$. Unlike [19], it is obvious that this XPGP-based model exploits attribute and relational information in preference learning, enabling score propagate through the network.

Algorithm 1: The AXPGP algorithm for link-based active preference learning

Input: X (entity attributes), R (links), $\mathcal{A}$ (the set of active pairs, one random pair at the beginning)

1 $m_0 = \mathbf{0}$; K_0 is computed with Eq. (6).
2 **for** $t = 1$ *to* T **do**
3 Approximate the likelihood distribution of the new active pair o_t (i.e., compute $\tilde{\Sigma}_t$ and $\tilde{\mu}_t$ with Eq. (22));
4 Update the posterior distribution of utilities for all entities given o_t (i.e., compute K_t and m_t with Eq. (24));
5 Compute expected loss for each candidate entity pair with Eq. (28) and pick the entity pair with the largest loss based on Eq. (29);
6 Label the chosen entity pair and add it to $\mathcal{A}$;

Output: m_T (mean) and K_T (covariance matrix)

However, using XPGPs naively for active exploration would scale as $\mathcal{O}(t^3)$, where t is the number of actively selected points so far. Say we have $t-1$ active data points. After selecting an additional data point, we have to invert the new covariance matrix among all t data points, which takes $\mathcal{O}(t^3)$ time. To meet the requirement on computational efficiency of active exploration, we propose an incremental update method for the XPGP model – the main contribution of the current paper. The incremental inference approach has scaling behavior comparable to the diagonal covariance method in [19]. We empirically evaluate the method using LETOR [18] benchmark datasets. The results show that AXPGP learns substantially faster than algorithms that cannot model dependencies.

The rest of the paper is organized as follows. We start off by introducing the link-based active preference learning method AXPGP in Sec. 2. We then discuss the model, incremental inference, and active learning. Before concluding, we present our experimental analysis.

2 Probabilistic Framework for Link-based Active Preference Learning

The active exploration model we assume can be outlined as follows. Denote with $\mathcal{T}$ the set of all possible entity preferences. Starting with an empty training set $\mathcal{A} = \emptyset$, we perform the following steps at each iteration: (1) a score (i.e. expected loss) is computed for all pairs in $\mathcal{J} = \mathcal{T} \setminus \mathcal{A}$ based on the current distribution of the utilities; (2) the pair with the largest score is picked and added to the training set $\mathcal{A}$ with its true preference; (3) the distribution of the utilities is updated based on the new $\mathcal{A}$. Note that the number of all possible entity preferences, i.e., the size of $\mathcal{T}$ is $\mathcal{O}(n^2)$ — much larger than the number n of entities.

The procedure simulates the interactions between users and information-retrieval systems and is summarized in Alg. 1. Before providing more details, let

us introduce some notations. We assume that there are (1) a set of n entities $E = \{e_1, \ldots, e_n\}$ with attributes $X = \{x_i : x_i \in \mathbb{R}^D, i = 1, \ldots, n\}$, (2) relations $R = \{r_{i,j} : i, j \in 1, \ldots, n\}$ among the entities, and (3) a set of m observed preferences (i.e., pairwise rankings) among entities, $O = \{e_{i_s} \succ e_{j_s} : s = 1, \ldots, m; i_s, j_s \in 1, \ldots, n\}$ (i_s and j_s are entities involved in the s-th observed preference). With r_i, we will denote all relations in which entity e_i participates.

2.1 Link-based Preference Learning with Gaussian Processes

Gaussian process[3] (GP) models [20] are powerful nonparametric tools for Bayesian supervised learning. In contrast to other kernel machines such as support vector machines, GPs are probabilistic models, which means that they yield "error bars" on predictions and allow standard Bayesian model selection to be used. They generalize multivariate Gaussian distributions over finite dimensional vectors to infinite dimensionality. Specifically, a GP defines a distribution over functions, i.e. each draw from a Gaussian process is a function, and it is completely characterized by its mean function $m(x) := [f(x)]$ and covariance (kernel) function $C(x, x') := [(f(x) - m(x))(f(x') - m(x'))]$. One attractive property of GPs is that any finite set of function values $f = \{f(x_1), \ldots, f(x_n)\}$ follow a multivariate Gaussian distribution so that mean and covariance can be computed based on the corresponding attribute vectors $x = \{x_1, \ldots, x_n\}$ with respect to the mean function and covariance functions.

Following the link-based GP (XPGP) model, the utility of each entity is modeled as a continuous latent variable, consisting of two latent function values $f(x_i)$ and $g(r_i)$ (shortened as f_i and g_i). $f(\cdot)$ and $g(\cdot)$ are functions of attributes and links, respectively. We define GP priors for the attribute-wise and for the link-wise latent function values. Specifically, we assume an infinite number of latent function values $\{f_1, f_2, \ldots\}$ follow a Gaussian process prior with mean function $m_a(x_i)$ and covariance function $k_a(x_i, x_j)$. Without loss of generality, we assume zero mean [20] so that the GP is completely specified by the covariance function only. Here we used the subscript a to emphasize that they are attribute-wise. In turn, any finite set of function values $\{f_i : i = 1, \ldots, n\}$ has a multivariate Gaussian distribution with mean zero and covariance matrix defined in terms of the covariance function of the GP. Formally, for the set of n entities, the prior distribution of the attribute-wise function values $f = (f_1, \ldots, f_n)^T$ can be represented as

$$P(f|X) = \frac{1}{(2\pi)^{\frac{n}{2}}|K|^{\frac{1}{2}}} \exp\left(-\frac{f^T K^{-1} f}{2}\right). \quad (1)$$

[3] Several GP models for preference learning have been proposed. For example Chu and Ghahramani [4] also considered the entity ranking problem (i.e. the setting discussed here) by introducing a novel likelihood function to express the entity-wise ordinal information. As another example, Guiver and Snelson [11] recently presented a sparse GP model for soft ranking problem for large-scale datasets. All previous GP models are reported to provide good performance on real-world datasets but they do not consider relational information.

Here, K denotes the $n \times n$ covariance matrix whose ij-th entry is computed in terms of the covariance function with the corresponding attributes x_i and x_j. The covariance function can be any Mercer kernel function, e.g. squared exponential (SE)

$$k(x_i, x_j) = \kappa^2 \exp(-\frac{\rho^2}{2} \sum\nolimits_d^D (x_{i,d} - x_{j,d})^2), \tag{2}$$

where κ and ρ are parameters of the covariance function, and $x_{i,d}$ denotes the d-th dimension of the attribute x_i.

Similarly, we place a zero-mean GP over $\{g_1, g_2, \ldots\}$. Again, $\{g_i : i = 1, \ldots, n\}$ follow a multivariate Gaussian distribution with mean zero and covariance matrix K_r. Since entities and links form a graph, we can naturally employ graph-based kernels to obtain the covariances, see e.g. [23]. The simplest graph kernel might be the regularized Laplacian

$$K_r = [\beta(\Delta + I/\iota^2)]^{-1}, \tag{3}$$

where β and ι are two parameters of the graph kernel. Δ denotes the combinatorial Laplacian, which is computed as $\Delta = D - W$, where W denotes the adjacency matrix of a weighted, undirected graph, i.e., $W_{i,j}$ is taken to be the weight associated with the edge between i and j. D is a diagonal matrix with entries $d_{i,i} = \sum_j w_{i,j}$. Formally, the prior distribution of the link-wise function values $g = (g_1, \ldots, g_n)^T$ is: $P(g|R) = \mathcal{N}(0, K_r)$

$$= \frac{1}{(2\pi)^{\frac{n}{2}} |K_r|^{\frac{1}{2}}} \exp\left(-\frac{g^T K_r^{-1} g}{2}\right) . \tag{4}$$

The utility ξ_i of an entity is a linear combination of the two function values:

$$\xi_i = \omega_1 f_i + \omega_2 g_i. \tag{5}$$

Since f_i and g_i have GP priors, their weighted sum ξ_i also follows a GP prior. For a finite number of entities, the prior is again reduced to a multivariate Gaussian distribution with mean zero and covariance matrix K, that is

$$P(\xi|X, R) = \mathcal{N}(0, K); \quad K = \omega_1^2 K_a + \omega_2^2 K_r. \tag{6}$$

After defining the prior of utilities, we now need to model how the utilities probabilistically decide the preferences, i.e. the likelihood distribution. For a preference o involving the entities e_i and e_j, $P(o_s|\xi_i, \xi_j)$ is defined as [4]

$$\int_{-\infty}^{\xi_i - \xi_j} \mathcal{N}(t|0, 1)dt \equiv \Phi(\xi_i - \xi_j) . \tag{7}$$

The likelihood is a cumulative Gaussian. This encodes the natural assumption: *The larger the difference between the utilities of e_i and e_j, the more likely is it that e_i is preferred to e_j.*

Finally, the joint probability of m observed preferences and n utilities $\xi = \{\xi_1, \ldots, \xi_n\}$ can be written as

$$P(\xi, O|X, R) = \mathcal{N}(\xi|0, K) \prod_{s=1}^{m} \Phi(\xi_{i_s} - \xi_{j_s}). \tag{8}$$

In this link-based GP ranking framework, the utilities are dependent on each other, meaning that knowing the preference of two entities can not only improve our utility predictions of the two entities, but also that of other entities which are not involved in the preference. Thus the ranking of a whole set of entities can be refined. How to diffuse the preference information among entities is decided by their dependencies. The probabilistic dependencies between them are captured with the covariance $cov(\xi_i, \xi_j)$, i.e. the entry (i, j) in the covariance matrix K, which measures to what extent the utilities change together. If two entities are inter-linked or have similar attributes, then their covariance $cov(\xi_i, \xi_j)$ is high. Roughly speaking, if ξ_i is high, then so is ξ_j, and vice versa.

2.2 Incremental Inference Method for AXPGP

Assume that in the first t iterations, we collect t preferences $\{o_1, \ldots, o_t\}$ and have learned the posterior distribution $P(\xi|X, R, o_1, \ldots, o_t)$. At iteration $t+1$, a new preference o_{t+1} is collected, and we need to compute:

$$P(\xi|X, R, o_1, \ldots, o_{t+1}) = \frac{P(\xi|X, R)P(o_1|\xi) \cdots P(o_t|\xi)P(o_{t+1}|\xi)}{P(o_1, \ldots, o_t, o_{t+1}|X, R)}. \tag{9}$$

That means we have to completely re-compute the posterior based on all preferences collected so far. Then a question naturally arises: could we exploit the previous computation and only focus on the new preference. To meet the challenge, we develop an incremental inference method. Now the posterior distribution is represented as

$$P(\xi|X, R, o_1, \ldots, o_{t+1}) = \frac{P(\xi|X, R, o_1, \ldots, o_t)P(o_{t+1}|\xi)}{P(o_{t+1}|X, R, o_1, \ldots, o_t)}, \tag{10}$$

where the first term in the numerator is obtained from the last iteration, and can be viewed as the *learned prior* of the current iteration. It is approximated with a Gaussian distribution with mean and covariance matrix denoted as m_t and K_t. Thus we have

$$P(\xi|X, R, o_1, \ldots, o_t)P(o_{t+1}|\xi) \approx \mathcal{N}(\xi|m_t, K_t)\Phi(o_{t+1}|\xi) \ .$$

Since the likelihood is not conjugate with the prior, an analytical solution is intractable. To solve the inference problem, the expectation propagation (EP) algorithm [16] is used. Namely, we use an unnormalized Gaussian distribution

$$\tilde{Z}_{t+1}\mathcal{N}(\xi_{i_{t+1}}, \xi_{j_{t+1}}|\tilde{\mu}_{t+1}, \tilde{\Sigma}_{t+1}) \tag{11}$$

to approximate the likelihood $\Phi(o_{t+1}|\xi)$. Note that (11) is a 2-dimensional Gaussian. Now the inference problem (10) is reduced to an optimization problem:

$$\begin{aligned}&\mathcal{N}(\xi_{i_{t+1}},\xi_{j_{t+1}}|\mu_t,\Sigma_t)\Phi(o_{t+1}|\xi_{i_{t+1}},\xi_{j_{t+1}})\\&\quad\leftarrow\mathcal{N}(\xi_{i_{t+1}},\xi_{j_{t+1}}|\mu_t,\Sigma_t)\tilde{Z}_{t+1}\mathcal{N}(\xi_{i_{t+1}},\xi_{j_{t+1}}|\tilde{\mu}_{t+1},\tilde{\Sigma}_{t+1}),\end{aligned}\tag{12}$$

where μ_t and Σ_t denote the entries in m_t and K_t corresponding to the entities $e_{i_{t+1}}$ and $e_{j_{t+1}}$ involved in the preference o_{t+1}. The product on the right side equals to $\mathcal{N}(\xi_{i_{t+1}},\xi_{j_{t+1}}|\hat{\mu},\hat{\Sigma})\ \tilde{Z}_{t+1}/C$, where

$$\begin{aligned}&\hat{\mu}=\hat{\Sigma}(\Sigma_t^{-1}\mu_t+\tilde{\Sigma}_{t+1}^{-1}\tilde{\mu}_{t+1}),\qquad\hat{\Sigma}=(\Sigma_t^{-1}+\tilde{\Sigma}_{t+1}^{-1})^{-1},\\&C=2\pi|\Sigma_t+\tilde{\Sigma}_{t+1}|^{\frac{1}{2}}\cdot\exp\left(\frac{1}{2}(\mu_t-\tilde{\mu}_{t+1})^T(\Sigma_t+\tilde{\Sigma}_{t+1})^{-1}(\mu_t-\tilde{\mu}_{t+1})\right).\end{aligned}\tag{13}$$

Thus the inference problem is reduced again: optimizing $\hat{\mu}$ and $\hat{\Sigma}$ to make the right side close to the left side, and then deriving $\tilde{\mu}_{t+1}$ and $\tilde{\Sigma}_{t+1}$ based on (13). To satisfy (12), we only need to match their first and second moments [20]. Since the product on the right side is an unnormalized Gaussian, we need to impose an additional condition, i.e. that the zero-th moments should also match. The zero-th moment of the left side is $\int\mathcal{N}(\xi_{i_{t+1}},\xi_{j_{t+1}}|\mu_t,\Sigma_t)\Phi(o_{t+1}|\xi_{i_{t+1}},\xi_{j_{t+1}})$

$$=\int_{-\infty}^{\varrho^T\mu_t}\mathcal{N}(a|0,1+\varrho^T\Sigma_t\varrho)da=\Phi\left(\frac{\varrho^T\mu_t}{\sqrt{1+\varrho^T\Sigma_t\varrho}}\right),\tag{14}$$

where ϱ denotes $y_{t+1}[1,-1]^T$. The term y_{t+1} is 1 if $e_{i_{t+1}}\succ e_{j_{t+1}}$, and -1 otherwise. Since the zero-th moments of the two sides are equal, we have

$$\tilde{Z}_{t+1}=C\ \Phi\left(\frac{\varrho^T\mu_t}{\sqrt{1+\varrho^T\Sigma_t\varrho}}\right).\tag{15}$$

The first moment of the left side is

$$\begin{aligned}&\frac{\partial}{\partial\mu_t}\int\mathcal{N}(\xi_{i_{t+1}},\xi_{j_{t+1}}|\mu_t,\Sigma_t)\Phi(o_{t+1}|\xi_{i_{t+1}},\xi_{j_{t+1}})\\&\quad=\int\mathcal{N}(\xi_{i_{t+1}},\xi_{j_{t+1}}|\mu_t,\Sigma_t)\Phi\left(\chi^T\varrho\right)(\chi-\mu_t)^T\Sigma_t^{-1},\end{aligned}\tag{16}$$

where χ^T denotes the vector $\left[\xi_{i_{t+1}},\xi_{j_{t+1}}\right]$. The first moment of the right side is

$$\frac{\partial}{\partial\mu_t}\Phi(\frac{\varrho^T\mu_t}{\sqrt{1+\varrho^T\Sigma_t}})=\frac{\varrho^T\mu_t}{\sqrt{1+\varrho^T\Sigma_t}}\mathcal{N}\left(\frac{\varrho^T\mu_t}{\sqrt{1+\varrho^T\Sigma_t}}\right)\tag{17}$$

Making the two first moments equal, we obtain:

$$\begin{aligned}&\hat{\mu}=\mathbb{E}\left(\chi\right)=\mu_t+\frac{S}{Z\sqrt{1+\varrho^T\Sigma_t\varrho}}\Sigma_t\varrho,\\&z=\frac{\varrho^T\mu_t}{\sqrt{1+\varrho^T\Sigma_t\varrho}};\quad Z=\Phi\left(z\right);\quad S=\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{z^2}{2}\right).\end{aligned}\tag{18}$$

Now we compute the second moment of the left side:

$$\frac{\partial^2}{\partial \mu_t^2} \int \mathcal{N}(\xi_{i_{t+1}}, \xi_{j_{t+1}} | \mu_t, \Sigma_t) \Phi(o_{t+1} | \xi_{i_{t+1}}, \xi_{j_{t+1}}) \\ = Z\, \mathbb{E}\left[\Sigma_t^{-1}(\chi - \mu_t)(\chi - \mu_t)^T \Sigma_t^{-1}\right] - Z\, \Sigma_t^{-1}. \tag{19}$$

and that of the right side:

$$\frac{\partial^2}{\partial \mu_t^2} \Phi\left(\frac{\varrho^T \mu_t}{\sqrt{1 + \varrho^T \Sigma_t}}\right) = -\varrho\varrho^T \frac{zS}{1 + \varrho^T \Sigma_t \varrho}\,. \tag{20}$$

Since the two moments are equal, we obtain $\mathbb{E}(\chi\chi^T) =:$

$$\frac{zS}{(1 + \varrho^T \Sigma_t \varrho) Z} \Sigma_t \varrho\varrho^T \Sigma_t + \Sigma_t + \mathbb{E}(\chi)\mu_t^T + \mu_t \mathbb{E}(\chi^T) - \mu_t \mu_t^T. \tag{21}$$

Now, the second central moment $\hat{\Sigma}$ can be computed with $\mathbb{E}(\chi\chi^T) - \mathbb{E}(\chi)\mathbb{E}(\chi^T)$. Substituting (18) and (21), we can get $\hat{\Sigma}$. In summary, the moment matching procedure yields the following equations to compute $\tilde{\mu}_{t+1}$, $\tilde{\Sigma}_{t+1}$:

$$\begin{aligned} &z = \frac{\varrho^T \mu_t}{\sqrt{1 + \varrho^T \Sigma_t \varrho}}; && S = \frac{1}{\sqrt{2\pi}} \exp(-\frac{z^2}{2}); \quad Z = \Phi(z); \\ &\hat{\mu} = \mu_t + \frac{S \Sigma_t \varrho}{Z \sqrt{1 + \varrho^T \Sigma_t \varrho}}; && \hat{\Sigma} = \Sigma_t - \frac{zSZ + S^2}{Z^2(1 + \varrho^T \Sigma_t \varrho)} \Sigma_t \varrho\varrho^T \Sigma_t; \\ &\tilde{\Sigma}_{t+1} = (\hat{\Sigma}^{-1} - \Sigma_t^{-1})^{-1}; && \tilde{\mu}_{t+1} = \tilde{\Sigma}_{t+1}(\hat{\Sigma}^{-1}\hat{\mu} - \Sigma_t^{-1}\mu_t). \end{aligned} \tag{22}$$

After getting the approximate likelihood, we can compute the posterior distribution of utilities. Since all entities are dependent on each other in the AXPGP algorithm, the update of the utility of one entity will change the utilities of all others. Thus, we need to compute the *new* distribution of all ξ_i at the iteration $t+1$, i.e., the posterior distribution Eq. (10). The product of two Gaussian is an unnormalized Gaussian, so $P(\xi | X, R, o_1, \ldots, o_t, o_{t+1})$ is approximated with:

$$\begin{aligned} \frac{\mathcal{N}(\xi | m_t, K_t) \tilde{Z}_{t+1} \mathcal{N}(\xi_{i_{t+1}}, \xi_{j_{t+1}} | \tilde{\mu}_{t+1}, \tilde{\Sigma}_{t+1})}{P(o_{t+1} | X, R, o_1, \ldots, o_t)} &= \mathcal{N}(\xi | m_{t+1}, K_{t+1}) \\ K_{t+1} &= (K_t^{-1} + [\hat{\imath}, \hat{\jmath}]\ \tilde{\Sigma}_{t+1}^{-1} [\hat{\imath}, \hat{\jmath}]^T)^{-1}; \\ m_{t+1} &= K_{t+1}(K_t^{-1} m_t + [\hat{\imath}, \hat{\jmath}]\ \tilde{\Sigma}_{t+1}^{-1} \tilde{\mu}_{t+1}). \end{aligned} \tag{23}$$

Here, $[\hat{\imath}, \hat{\jmath}]$ denotes the unit vectors, which is one at the entry i_{t+1} (resp. j_{t+1}) and zero otherwise. Note that this approximate distribution is the learned prior of the next iteration $t+2$. Given a new preference o_{t+2}, we can update the distribution $P(\xi | X, R, o_1, \ldots, o_t, o_{t+1}, o_{t+2})$ via (22) and (23) efficiently. Because the information from the historical data $\{o_1, \ldots, o_t\}$ propagates to the new iteration $t+1$ via the *learned prior* $P(\xi | X, R, o_1, \ldots, o_t)$, the new iteration only needs to re-compute the distribution over utilities by one single new observation

o_{t+1}, rather than to re-compute the distribution conditioned on all observations. Thus, it implements an incremental inference method.

The computation in (22) only involves operations on two-dimensional matrices which is cheap to compute. (23), however requires to compute the inverse of a $n \times n$ matrix, which scales $O(n^3)$. The cost, however, can be reduced by using the Woodbury, Sherman & Morrison formula yielding a rank-two algorithm:

$$K_{t+1} = K_t - ABA^T; \quad A = K_t[\hat{\imath}, \hat{\jmath}]; \quad B^{-1} = \left(\tilde{\Sigma}_{t+1} + \Sigma_t\right)$$
$$m_{t+1} = m_t - AB\left(\mu_t + \Sigma_t \tilde{\Sigma}_{t+1}^{-1} \tilde{\mu}_{t+1}\right) + A\tilde{\Sigma}_{t+1}^{-1}\tilde{\mu}_{t+1}, \tag{24}$$

where A denotes the columns of K_t corresponding to the entities $e_{i_{t+1}}$ and $e_{j_{t+1}}$ involved in the preference o_{t+1}. In comparison with the Glicko [19] method, the information of the preference o_{t+1} is propagated to all the entities based on the dependencies between them. For example, the change of the covariance matrix is the product ABA^T, where B represents the change of covariances of the entities $e_{i_{t+1}}$ and $e_{j_{t+1}}$ introduced by the new observation. A represents the covariances between the two entities and all other unrelated entities to the new observation, by which the information propagates.

Here, we consider inference only in a transductive setting. However, the proposed method can also be extended to an inductive setting. The key challenge is getting the covariance between the new entities and the known ones based on relations. This can be addressed via several possible methods: Nyström [22] and similarity matching [28].

2.3 Active Exploration

There are two major characteristics, which distinguish the active exploration for preference learning from the ones for classification/regression: (1) preferences are pairwise (or list-wise) rather than entity-wise; (2) the entity property *utility* that decides the preference is latent and unobservable. Even if we observe the preferences of entity pairs, the utilities are still unknown. Now that we are armed with the posterior distribution of the latent utilities, we can use it to decide which is the next entity pair to query based on the expected loss.

Let ξ^* denote the current utility estimates used to produce the predicted ranking of items in the current iteration. Furthermore, let $\mathcal{L}(\xi^*, \xi)$ be the loss incurred by the estimates ξ^* compared to the true utilities ξ. Since the true utilities are unknown and random, the value of the loss function is also random. We thus use the expectation of the loss functions with respect to the distribution of the true utilities. Suppose the loss function is pairwise decomposable (i.e. a sum of independent pairwise losses), then the total expected loss over all pairs can be factorized to

$$\mathbb{E}_{P(\xi|O)}\left[\mathcal{L}(\xi^*, \xi)\right] = \sum\nolimits_{i=1}^{N} \sum\nolimits_{j=i+1}^{N} \mathbb{E}_{P(\xi_i, \xi_j|O)}\left[\mathcal{L}(\xi_i^*, \xi_j^*, \xi_i, \xi_j)\right]. \tag{25}$$

Based on [19], the mode (i.e., mean μ) of their approximate posterior is used as the estimate ξ^*. [19] proves that for many loss functions, sorting entities by the

mode results in the ranking that minimizes the expected loss Eq. (25). Radlinski and Joachims [19] proposed the following loss function for information retrieval problems, since it (a) models misorderings, (b) takes into account that accuracy is more important at the top of the ranking, and (c) treats errors on pairs with almost equal utility different from pairs with large utility gap. In particular, for a pair of entities e_i and e_j,

$$\mathcal{L}(\mu_i, \mu_j, \xi_i, \xi_j) = e^{-\gamma_{ij}} \left((\mu_i - \mu_j) - (\xi_i - \xi_j)\right)^2 \mathbb{1}_{misordered}, \tag{26}$$

where γ_{ij} denotes the minimum rank of e_i and e_j when all entities are ranked by their estimates. The term $e^{-\gamma_{ij}}$ emphasizes the importance of entities ranked higher in the list. The higher the ranks of the two entities, the larger the loss if the estimated preference is wrong. The term $\mathbb{1}_{misordered}$ is a function taking value one if $(\mu_i - \mu_j)(\xi_i - \xi_j) < 0$, and zero otherwise. Namely, we consider the difference between the estimations and the true values as loss only when it is substantial enough to influence the preference of the two entities. The expectation of the loss function (26) is computed as follows. Let δ_{ij} denote $\xi_i - \xi_j$. Since ξ_i and ξ_j are Gaussian, their difference δ_{ij} is still Gaussian,

$$\delta_{ij} \sim N(\delta_{ij} | \hat{\delta}_{ij}, \nu_{ij}^2) \,, \tag{27}$$

with $\hat{\delta}_{ij} = \mu_i - \mu_j$ and $\nu_{ij}^2 = var(\xi_i) + var(\xi_j) - 2cov(\xi_i, \xi_j)$, where $var(x)$ and $cov(x, y)$ denote the variance of x and the covariance between x and y. Permuting entities in each pair such that $\hat{\delta}_{ij}$ is always negative, we have the expectation:

$$\mathbb{E}_{P(\xi_i, \xi_j | O)} \left[\mathcal{L}(\mu_i, \mu_j, \xi_i, \xi_j)\right] = e^{-\gamma_{ij}} \left[\frac{\nu_{ij}^2}{2}\left(1 + erf\left(\frac{\hat{\delta}_{ij}}{\sqrt{2}\nu_{ij}}\right)\right) - \frac{\hat{\delta}_{ij}\nu_{ij}}{\sqrt{2\pi}} \exp\left(\frac{-\hat{\delta}_{ij}^2}{2\nu_{ij}^2}\right)\right]. \tag{28}$$

Radlinski and Joachims [19] introduced several exploration strategies, i.e., how to select the most informative entity pair based on the loss function. Here, we focus on the *largest expected loss pair (LELpair)*, which selects the entity pair e_i and e_j that has the largest pairwise expected loss contribution, i.e.,

$$\arg\max_{i \neq j} \mathbb{E}_{P(\xi_i, \xi_j | O)} \left[\mathcal{L}(\mu_i, \mu_j, \xi_i, \xi_j)\right] \tag{29}$$

We select the loss function (28) and the active strategy (29) due to their good performance and low computational cost. For more details on the theoretical and empirical analysis we refer to [19].

2.4 An Illustration

We have described the link-based active preference learning method AXPGP. To illustrate the principle behind AXPGP, we visualize the procedure with a simple example. Assume that there are six papers $\{a, b, c, d, e, f\}$ ranked as $a \succ b \succ c \succ d \succ e \succ f$. We know that for (a, b) and (d, e) co-author relations

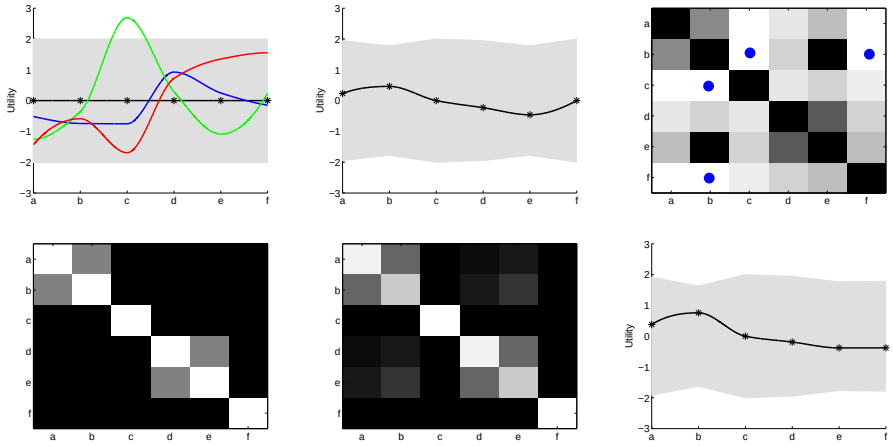

Fig. 1. Illustration of the AXPGP algorithm. Top-left: Three utility functions drawn at random from a GP prior. The shaded area shows the 95% confidence region. Bottom-left: the prior covariance matrix. The smaller the covariance is, the darker the cell is. Top-middle: the posterior utilities learned with the prior and the observed preference $b \succ e$. The line is the posterior mean. Bottom-middle: The posterior covariance matrix. Top-right: The expected loss of each entity pair. The smaller the loss is, the darker the cell is. The blue points specify the pairs with the largest loss. Bottom-right: The updated posterior utilities given the actively collected preference $b \succ f$.

exists. Intuitively, the utilities of a and b resp. d and e should be correlated. Formally, we compute the covariance matrix using the graph kernel (3), shown in Fig 1 bottom-left, which just verifies our intuition. Since there is no preference information given, the expected utilities are the same for all entities (Fig. 1 top-left). Assume now that a user tells us that she prefers b to e. Our belief on the utilities changes, and we compute the posterior distribution based on (23). Fig. 1 top-middle clearly shows that the expected utilities of b and e reflect this. Additionally, the expected utilities of a and d also change. Since there is co-author relation between them, the preference information propagates through the network to other entities. Now, we compute the loss using (28) and select the next entity pair to ask for a label using (29). Fig. 1 top-right shows the loss for each entity pair. One can see that the pairs $\{(a,d),(a,e),(b,d)\}$ have the smaller expected losses. The reason is again that a and d are related to b and e. The interesting fact, however, is the two isolated entities (c,f) do not give the largest loss. The reason is that, given loss function (26), it is more informative to find out whether c and f beat the current leader than learning about the relation between the two. Randomly selecting one pair (b,f) with the largest loss, the updated utilities are shown in Fig. 1 bottom-right. At this point, the ranking of the whole set of entities is already very close to the actual one except for the order of a and b. That is reasonable, because so far there is no explicit

or implicit information available on their preference. The order between a and $\{c, d, e, f\}$, however, can already be accurately estimated.

3 Empirical Analysis

We evaluate the link-based active preference learning method AXPGP on two real-world datasets, namely OHSUMED and TREC in the LETOR repository [18]. The AXPGP method is compared with the Glicko [19] method to evaluate whether the relational model speeds up active learning. Furthermore, we compare computational efficiency, and evaluate in how far active learning using the AXPGP outperforms random sampling.

Mean average precision (MAP) is used as the performance measure [15]. MAP is defined as the average AP over multiple rankings, $AP = \frac{1}{N^*} \sum_{n=1}^{N} prec(n)\delta(n)$, where N^* denotes the number of relevant documents for the query. $\delta(n)$ equals to one if the n-th document in the ranking is relevant, and zeros otherwise. $prec(n) = \frac{1}{n} \sum_{i=1}^{n} \delta(i)$ is the precision after observing the first n documents in the ranking.

In each iteration of the following experiments, the algorithm asks for feedback on the preference of one entity pair, e_i and e_j. The feedback label is generated according to the order of the true utilities ξ_i and ξ_j. After observing a preference, the algorithm updates the model and we compute the MAP over all documents. To abstract from randomness in breaking ties of the LELpair criterion, each experiment is repeated 10 times. We used (2) and (3) to capture the feature and relational information.

3.1 Data Description

LETOR is a benchmark dataset for learning to rank in information retrieval. We perform experiments on two datasets in LETOR: one is the OHSUMED dataset about medical articles, the other is the TREC dataset about .gov webpages.

In the OHSUMED dataset, each document consists of title, abstract, MeSH indexing terms, author, source, and publication type. There are 106 queries for which manual relevance judgments on the scale 'definitely relevant' (2), 'partially relevant' (1), and 'not relevant' (0) are available. We follow the experiment setup in [19]. In particular, since such coarse judgments are unrealistic in many real-world applications, we break ties by adding uniform noise in the range $[-0.5, 0.5]$ to the true relevance degrees. Note that this preserves the relative order between definitely relevant (resp. partially relevant) documents and partially relevant (resp. not relevant) ones, but breaks ties within each relevance level. The dataset contains a feature vector for each query-document pair describing the match of the document to the query. Furthermore, we make use of the same relational information that was previously exploited in [17]. The relations are based on similarities, i.e. there is a weighted complete graph between documents. On average, there are about 152 documents per query.

The TREC dataset was originally collected for a special track on web information retrieval at TREC 2004. The goal of the track was to explore the performance of retrieval methods on large-scale data with hyperlinked structure such as the World Wide Web. The data was crawled from .gov domains in January, 2002. In total, there are 1,053,110 html documents with 11,164,829 hyperlinks. To each query, documents were assigned labels by human experts. Each document has two possible states: relevant(1)/irrelevant(0). Qin *et al.* [18] processed the TREC dataset and turned it into a benchmark for information retrieval. Again, we add uniform noise in the range $[-0.5, 0.5]$ to break ties and simulate a realistic situation in many real-world applications. On average, there were about 1000 documents per query which are linked with about 2387 hyperlinks.

3.2 Experimental Results

Can the AXPGP Exploit Cross-Document Information? We first compare the AXPGP to Glicko following [19]. In particular, we evaluate whether the AXPGP can effectively transfer information between documents by using the features and the relations, while Glicko needs to learn each utility ξ_i individually. We run Glicko and AXPGP with the incremental updates separately for each individual query in the OSHUMED data. The MAP performance of the two methods averaged over all single-query experiments is given in the top-left plot of Fig. 2. The AXPGP does indeed learn significantly (according to both a Wilcoxon rank sum test and a t-test at each multiple of 100 iterations) faster than Glicko. Furthermore, the variability of the AP across multiple queries is substantially lower for the AXPGP (Fig. 2, top middle) than for Glicko (Fig. 2, right), especially for large numbers of iterations. The results on the TREC data, averaged over 10 randomly selected queries, show the similar trends.

How Efficient is the AXPGP Compared to Glicko? The left-hand plots of Fig. 3 show the average CPU time it takes to learn for a certain number of iterations. While Glicko is the fastest in absolute time, the incremental update shows the same scaling and is slower only by a constant factor. The incremental update is substantially faster than the full update and shows better scaling behavior. It remains to investigate whether the incremental update is less accurate than the full update. The right-hand plots of Fig. 3 show that the MAP performance of the incremental update is not substantially lower than the MAP of the full update.

How Large is the Benefit of Active Learning? Finally, we evaluate in how far the active selection strategy improves on randomly selecting training pairs under the GP model. Here, we learn a single model over multiple queries using the feature-vector representation to define the covariance matrix of the GP. Fig. 4 (left) shows that the MAP grows significantly and substantially faster when using active learning. Furthermore, variability over multiple runs of the learning algorithms is much smaller for active learning (Fig. 4, middle) than for random sampling (Fig. 4, right).

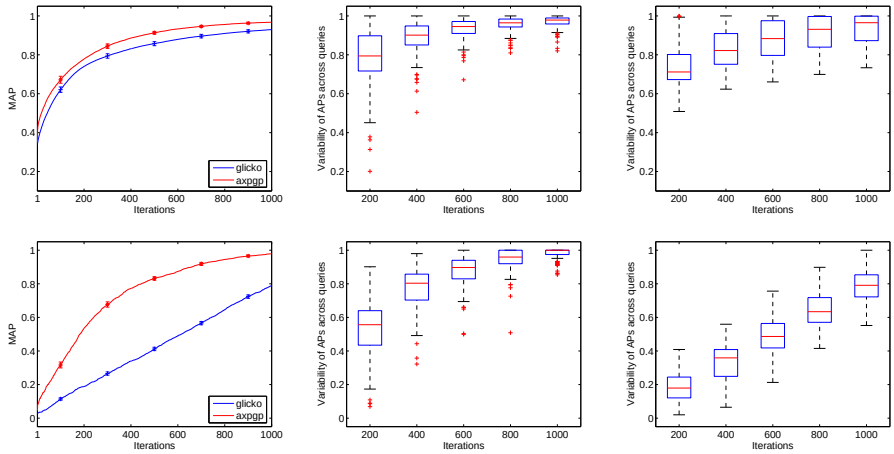

Fig. 2. MAP on the OHSUMED (top) and TREC (bottom) datasets when learning a separate model for each query. The average performance (with standard error) is given in the left-hand plot. The box plots show variability across queries for the AXPGP (middle) and Glicko (right).

4 Conclusions

In this paper, we explored how to integrate active learning into preference learning methods that can model dependencies from both feature-vector representations as well as relations. On real-world datasets for information retrieval, we have shown that actively learning a link-based Gaussian process ranking model is substantially faster than algorithms that cannot model dependencies. The key to the computational efficiency is a novel incremental update method that makes active exploration of link-based Gaussian process models essentially as fast as for the traditional models.

The most natural avenue for future work is the adaption of sparse Gaussian processes techniques to tackle large-scale datasets. In general, one should explore and develop active learning techniques for other relational models and tasks.

Acknowledgments This work was funded in part by the Fraunhofer ATTRACT Fellowship "Statistical Relational Activity Mining" (STREAM), and by the German Federal Ministry of Economy and Technology (BMWi) under the THESEUS project, as well as by the NSF Awards IIS-0905467 and IIS-0812091.

References

1. M. Bilgic and L. Getoor. Link-based active learning. In *Proceedings of NIPS Workshop on Analyzing Networks and Learning with Graphs*, 2009.
2. K. Brinker. Active learning of label ranking functions. In *Proc. of ICML*, 2004.

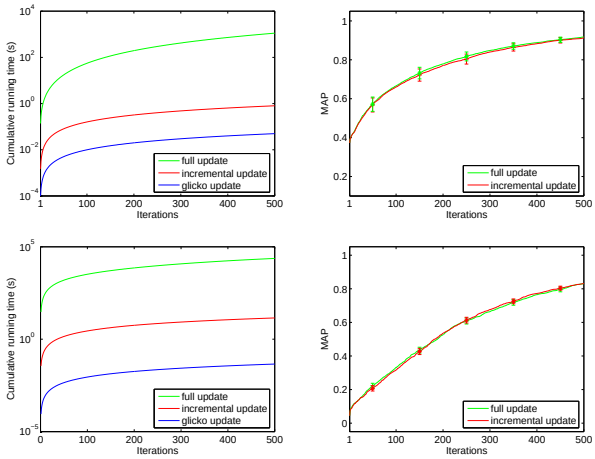

Fig. 3. Computational efficiency of the AXPGP with full and incremental updates compared to Glicko (left). Comparison of MAP for AXPGP with full and incremental updates (right). The OSUMED (top) and TREC (bottom) results are respectively averaged over 30 and 10 randomly selected queries for efficiency reasons.

3. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML*, 2005.
4. W. Chu and Z. Ghahramani. Preference learning with gaussian processes. In *ICML*, 2005.
5. P. Donmez and J. G. Carbonell. Active sampling for rank learning via optimizing the area under the roc curve. In *Proc. ECIR*, 2009.
6. N. d. E. Brochu and A. Ghosh. Active preference learning with discrete choice data. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.
7. Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *JMLR*, 4:933–969, 2003.
8. J. Fürnkranz and E. Hüllermeier. Preference learning and ranking by pairwise comparison. In *Preference Learning*. Springer-Verlag, 2010.
9. F. Geerts, H. Mannila, and E. Terzi. Relational link-based ranking. In *Proceedings of VLDB-04*, pages 552–563, 2004.
10. L. Getoor and B. Taskar, editors. *An Introduction to Statistical Relational Learning*. MIT Press, 2007.
11. J. Guiver and E. Snelson. Learning to rank with softrank and gaussian processes. In *SIGIR*, 2008.
12. E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16–17):1897–1916, 2008.
13. T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2), April 2007.
14. K. Kersting and Z. Xu. Learning preferences with hidden common cause relations. In *Preceding of ECML09*, 2009.
15. C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

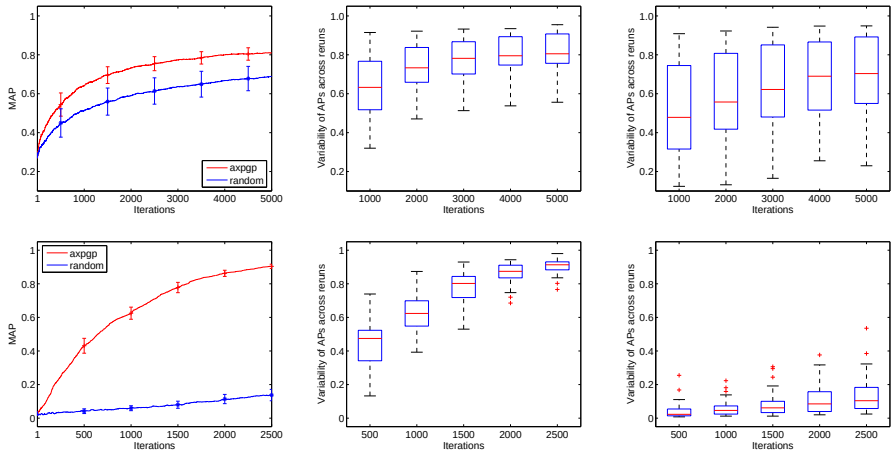

Fig. 4. Comparison of MAP (left), as well as variability when learning a single AXPGP model over multiple queries with active learning (middle) vs. random sampling (right). The top is the results on the OSHUMED with a random sample of 10 queries. The bottom is that on the TREC data with a random sample of 5 queries.

16. T. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT, 2001.
17. T. Qin, T. Liu, X. Zhang, D. Wang, W. Xiong, and H. Li. Learning to rank relational objects and its application to web search. In *WWW*, 2008.
18. T. Qin, T.-Y. Liu, J. Xu, and H. Li. Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval Journal*, 2010.
19. F. Radlinski and T. Joachims. Active exploration for learning rankings from clickthrough data. In *Proc. SIGKDD*, pages 570–579, 2007.
20. C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
21. M. Saar-Tsechansky and F. Provost. Active sampling for class probability estimation and ranking. *Machine Learning*, 54(2):153–178, 2004.
22. A. Schwaighofer, V. Tresp, and K. Yu. Learning gaussian process kernels via hierarchical bayes. In *NIPS 17*, 2005.
23. A. J. Smola and I. Kondor. Kernels and regularization on graphs. In *Annual Conference on Computational Learning Theory*, 2003.
24. B. Taskar, M. Wong, and D. Koller. Learning on the test data: Leveraging unseen features. In *ICML*, 2003.
25. H. Yu. Selective sampling for ranking with application to data retrieval. In *Proc. SIGKDD*, 2005.
26. Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. In *Conference on Learning Theory (COLT)*, 2009.
27. Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proc. ICML*, 2009.
28. X. Zhu, J. Lafferty, and Z. Ghahramani. Semi-supervised learning: From gaussian fields to gaussian processes. Technical Report CMU-CS-03-175, CMU, 2003.