Finding Convincing Arguments using Scalable Bayesian Preference Learning

First Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Second Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Abstract

We introduce a scalable Bayesian preference learning method for identifying convincing arguments in the absence of gold-standard ratings or rankings. In contrast to previous work, we avoid the need for separate approaches or pipelines to produce training data, predict rankings and perform pairwise classification. Although Bayesian methods are known to be effective when faced with sparse or noisy training data, they have not previously been used to identify convincing arguments. One deterrent is their perceived lack of scalability, which we address by developing a stochastic variational inference method for Gaussian process (GP) preference learning. We show how our method can be applied to predict argument convincingness from crowdsourced data, outperforming state-of-the-art methods, particularly when the data is sparse or noisy. We demonstrate how our Bayesian approach enables more effective active learning, thereby reducing the amount of data required to identify convincing arguments for new users and domains. While word embeddings are principally used with neural networks, our results show that word embeddings in combination with linguistic features also benefit GPs when predicting argument convincingness.

1 Introduction

Argumentation is intended to persuade the reader of a particular point of view and is an important way for humans to reason about controversial topics (Mercier and Sperber, 2011). The amount of argumentative text on any given topic can, however, overwhelm a reader, particularly considering the scale of historical text archives and the prevalence of social media platforms with millions of authors. To gain an understanding of a topic, it is therefore useful to identify high-quality, persuasive arguments from different sides of a debate.

Previously, Habernal and Gurevych (2016b) showed that it is possible to predict the convincingness of arguments taken from online discussion forums using models trained on one topic and transferred to another. In the absence of gold-standard ratings for the arguments, they crowdsourced pairwise preference labels indicating which argument in a pair the annotator found more convincing. As a means of eliciting convincingness, pairwise preferences have a number of advantages. Firstly, unlike ratings or scores, they do not require calibrating when annotators are biased toward high, low or middling scores. Pairwise comparisons are also more fine-grained than categorical labels and can lead to more reliable results with less cognitive burden on human annotators (Kendall, 1948; Kingsley, 2006). Implicit preferences can also be elicited from user actions, such as selecting a document from a list of summaries (Joachims, 2002).

In practice, however, preference data may be noisy, particularly if obtained from crowds or implicit feedback. We may also be faced with very small amounts of data when we move to new domains and topics, which can present a problem to methods such as deep neural networks (Srivastava et al., 2014). The approach used by Habernal and Gurevych (2016b) to handle unreliable crowd-

sourced data involved first determining consensus labels using the MACE algorithm (Hovy et al., 2013); these consensus labels were then used to train a classifier and as input to PageRank; the resulting rankings were then used as training data for regression models. However, such pipeline approaches can be prone to error propagation (Chen and Ng, 2016) and consensus algorithms such as MACE require multiple crowdsourced labels for each argument pair.

In contrast to previous work, we propose the use of preference learning techniques for argument convincingness to directly model the relationship between crowdsourced pairwise preferences and textual features, including word embeddings. We choose a Bayesian approach, since Bayesian methods have been to handle the problem of small datasets (e.g. (Xiong et al., 2011; Titov and Klementiev, 2012)), unreliable data (e.g. (Simpson et al., 2015)), and provide a good basis for active selection to reduce labelling costs (MacKay, 1992). Our method is based on the model of (Chu and Ghahramani, 2005), which assumes that preferences over items are described by a latent preference function. By providing a Gaussian process (GP) prior over this latent function, their method handles uncertainty in the function values due to noise and data sparsity in a principled manner. This approach has not previously been applied to text problems with large numbers of features and their proposed inference scheme was limited by a computational complexity of $\mathcal{O}(N^3)$, where N is the number of items. We address the problem of scalability by using recent advances in stochastic variational inference (SVI) (Hoffman et al., 2013) to develop a new inference approach for Gaussian process preference learning. We further show how we our method enables efficient optimisation of important hyper-parameters. We then demonstrate how our method can be applied to argument convincingness with a large number of linguistic features and high-dimensional text embeddings. Our evaluation compares Bayesian preference learning to established SVM and neural network approaches for predicting convincing arguments, and shows that our approach outperforms these alternatives, particularly with small and noisy datasets.

The rest of the paper is structured as follows. First, we review related work in more detail: on argumentation; Bayesian methods for preference learning; and scalable approximate inference. Section 3 then details the preference learning model, our SVI inference technique, and hyper-parameter optimisation method. Section 4 then presents a number of experiments: a comparison with the state-of-the art on predicting preferences in online debates; noisy datasets; active learning; and feature relevance determination. Finally, we present conclusions and avenues for future work.

2 Related Work

Recently, Habernal and Gurevych (2016a) analysed reasons provided by annotators for why one argument is more convincing than another. In this paper we assume that explicit reasons are not provided. Investigations by Lukin et. al (2017) demonstrated the effect of personality and the audience's prior stance on persuasiveness, although their work does not extend to modelling persuasiveness using preference learning. The sequence of arguments in a dialogue is another important factor in their ability to change the audience's opinions (Tan et al., 2016). Reinforcement learning has been used to choose the best argument to present to a user (Rosenfeld and Kraus, 2016; Monteserin and Amandi, 2013), but such approaches do not model user preferences for arguments with certain qualities.

The goal of learning an audience's preferences is to rank or score items or predict which item in a subset the user prefers. A preference for item x_i over x_j is written as $x_i \succ x_j$. Pairwise labels can be predicted using a generic classifier without the need to learn a total ordering. To do this, pairs of items are transformed either by concatenating the feature vectors of two items (Habernal and Gurevych, 2016b), or by computing the difference of the two feature vectors, as in SVM-Rank (Joachims, 2002). The classifier is then trained using the transformed feature vectors as input data and the preference labels binary class labels. However, the ranking of items is useful for producing ordered lists in response to a query – consider a sorted list of the most convincing arguments in favour of topic X. Another approach is to learn the ordering directly using Mallows models (Mallows, 1957), which define distributions over permutations of a list. Mallows models have been

extended to provide a generative model (Qin et al., 2010) and to be trained from pairwise preferences (Lu and Boutilier, 2011), but inference is typically costly since the number of possible permutations to be considered is $\mathcal{O}(N^2)$, where N is the number of items to be ranked. Modelling only the order of items means we are unable to quantify how closely rated items at similar ranks are to one another: how much better is the top ranked item from the second-rated?

To avoid the problems of classifier-based and permutation-based methods, another approach is to learn a set of real-valued scores from pairwise labels that can be used to predict rankings, pairwise labels, or as ratings for individual items. There are two established approaches for mapping discrete pairwise labels to real-valued scores: the Bradley-Terry-Plackett-Luce model (Bradley and Terry, 1952; Luce, 1959; Plackett, 1975) and the Thurstone-Mosteller model (Thurstone, 1927; Mosteller, 2006). More recently, Bayesian extensions of the Bradley-Terry-Plackett-Luce model were proposed by (Guiver and Snelson, 2009; Volkovs and Zemel, 2014), while the Thurstone-Mosteller model was used by (Chu and Ghahramani, 2005). This latter piece of work assumes a Gaussian process (GP) prior over the scores, which enables us to predict scores for previously unseen items given their features using a Bayesian nonparametric approach. Nonparametric methods allow the function complexity to grow with the amount of data. Gaussian processes are a well established tool for extrapolating from training data in a principled manner, taking into account model uncertainty that may arise when data for new domains is limited (Rasmussen and Williams, 2006).

The inference method used by Chu and Ghahramani (2005)has memory and computational costs that scale with $\mathcal{O}(N^3)$. Besides this limitation, there is also a computational and memory cost during training of $\mathcal{O}(N^2)$ due to the number of pairs in the training dataset. Recently, the stochastic variational inference (SVI) algorithm proposed by (Hoffman et al., 2013) has been used to address this problem in Gaussian process models (Hensman et al., 2013; Hensman et al., 2015) but has not previously been adapted for preference learning with GPs. The next section explains how we apply this technique to create a scalable preference learning method for argument convincingness.

3 Scalable Bayesian Preference Learning

Following Chu and Ghahramani (2005), we model the relationship between a latent preference function, f, and each observed pairwise label, $v_k \succ u_k$, where k is an index into a list of P pairs, as follows:

$$p(v_k \succ u_k | f(v_k), f(u_k), \delta_{v_k}, \delta_{u_k}) = \begin{cases} 1 & \text{if } f(v_k) + \delta_{v_k} \geq f(u_k) + \delta_{u_k} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\delta_i \sim \mathcal{N}(0, 1)$ is Gaussian-distributed noise. The noise term allows for variations in the observed preferences, which may occur if different annotators disagree or change their minds, or if the preferences are derived from noisy implicit data such as clicks streams. We deviate from Chu and Ghahramani (2005) by assuming δ_i has variance $\sigma = 1$, and instead scale the function f relative to this. This formulation is equivalent but is more convenient for variational inference. We marginalise the noise terms to obtain the preference likelihood:

$$p(v_k \succ u_k | f(v_k), f(u_k)) = \Phi\left(\frac{f(v_k) - f(u_k)}{\sqrt{2}}\right), \tag{2}$$

where Φ is the cumulative distribution function of a standard Gaussian distribution. For the latent function , f, we assume a Gaussian process prior: $f \sim \mathcal{GP}(0, k_\theta/s)$, where k_θ is a kernel function with hyper-parameters θ, and $s \sim \mathcal{G}(a_0, b_0)$ is an inverse scale parameter drawn from a gamma prior with shape a_0 and scale b_0. The kernel function controls the correlation between values of f at different points in the feature space.

The inference goal is to learn the posterior distribution over the function values $f(i)$ for each item i. Chu and Ghahramani (2005) used gradient descent to optimise a Laplace approximation. However, this approach produces a maximum a-posteriori (MAP) approximation, which takes the most probable values of parameters rather than integrating over their distributions in a Bayesian manner and has been

shown to perform poorly for tasks such as classification (Nickisch and Rasmussen, 2008). We address the desire for a better approximation by adapting a variational method based on the extended Kalman filter (EKF) (Reece et al., 2011; Steinberg and Bonilla, 2014) to the preference likelihood given by Equation 2. To make inference tractable, we approximate the preference likelihood using a Gaussian distribution: $p(v_k \succ u_k|f(v_k), f(u_k)) \approx \mathcal{N}(v_k \succ u_k; \Phi(\hat{f}_{v_k} - \hat{f}_{u_k}/\sqrt{2}), \nu_k)$. The variance v_k is estimated by moment matching with the variance of a beta posterior distribution $\mathcal{B}(p(v_k \succ u_k|f(v_k), f(u_k)); 1 + [v_k \succ u_k], 2 - v_k \succ u_k)$. The values of v_k for all pairs form a diagonal matrix $\boldsymbol{Q}$. This approximation means that the posterior distribution over f for items in the training set is also Gaussian: $p(f(\boldsymbol{x})|\boldsymbol{y}) \approx \mathcal{N}(f(\boldsymbol{x})|\hat{\boldsymbol{f}}, \boldsymbol{C})$ where $\boldsymbol{x}$ is a matrix of input features for the training items. Variational inference is then used to optimise the mean $\hat{\boldsymbol{f}}$ and covariance $\boldsymbol{C}$. by maximising a lower bound, $\mathcal{L}$, on the log marginal likelihood, $p(\boldsymbol{y}|\theta, a_0, b_0)$:

$$\begin{aligned}\mathcal{L}(q) \approx -\frac{1}{2}\{&L\log 2\pi + \log|\boldsymbol{Q}| - \log|\boldsymbol{C}| + \log|\boldsymbol{K}| \\ &+(\hat{\boldsymbol{f}}-\boldsymbol{\mu})\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}}-\boldsymbol{\mu}) \\ &+(\boldsymbol{y}-\Phi(\hat{\boldsymbol{z}}))^T\boldsymbol{Q}^{-1}(\boldsymbol{y}-\Phi(\hat{\boldsymbol{z}}))\} \\ &+\Gamma(a)-\Gamma(a_0)+a_0(\log b_0)+(a_0-a)\hat{\ln}\, s \\ &+(b-b_0)\hat{s}-a\log b, \end{aligned} \tag{3}$$

where L is the number of observed preference labels, $\boldsymbol{y} = [v_1 \succ u_1, ..., v_L \succ u_L]$ is a vector of binary preference labels, $\hat{\ln}\, s$ and $\hat{s}$ are expected values of s, and $\hat{\boldsymbol{z}} = \left\{\frac{\hat{f}_{v_k} - \hat{f}_{u_k}}{\sqrt{2}} \forall k = 1, ..., P\right\}$.

The variational approach described so far requires a scalable inference algorithm. We therefore adapt stochastic variational inference (SVI) (Hensman et al., 2013; Hensman et al., 2015) to preference learning. For SVI, we assume M *inducing points* with features $\boldsymbol{x}_m$. The inducing points act as a substitute for the complete set of feature vectors of the observed arguments, and allow us to choose $M << N$ to limit the computational and memory requirements. To choose representative inducing points, we use K-means to rapidly cluster the feature vectors, then used the cluster centres as inducing points. Given the inducing points, SVI further limits computational costs by using an iterative algorithm that only considers a subset of the data containing $P_n << P$ pairs at each iteration. The algorithm proceeds as follows:

1. Randomly initialise the mean at the inducing points, $\hat{\boldsymbol{f}}_m$, the covariance of the inducing points, $\boldsymbol{S}$, the inverse function scale expectations $\hat{s}$ and $\hat{\ln}\, s$, and the Jacobian of the pairwise label probabilities, $\boldsymbol{G}$.
2. Select a random subset of P_n pairwise labels.
3. Compute the mixing coefficient, $\rho_i = (n + \text{delay})^{-\text{forgetting}_\text{rate}}$, which controls the rate of change of the estimates, and the weight $w_n = \frac{P}{P_n}$, which weights each update according to the size of the random subsample.
4. Update each variable in turn using equations below.
5. Repeat from step 2 until convergence.
6. Use converged values to make predictions.

The equations for the updates at iteration n are as follows:

$$\begin{aligned}\boldsymbol{S}_n^{-1} = (1-\rho_n)\boldsymbol{S}_{n-1}^{-1} + \rho_n\big(&\hat{s}\boldsymbol{K}_{mm}^{-1} \\ &+w_n\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{nm}^T\boldsymbol{G}^T\boldsymbol{Q}^{-1}\boldsymbol{G}\boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-T}\big)\end{aligned} \tag{4}$$

$$\begin{aligned}\hat{\boldsymbol{f}}_{m,n} = \boldsymbol{S}_n\Big((1-\rho_n)\boldsymbol{S}_{n-1}^{-1}\hat{\boldsymbol{f}}_{m,n-1} + \rho_n w_n\boldsymbol{K}_{mm}^{-1} \\ \boldsymbol{K}_{nm}^T\boldsymbol{G}^T Q^{-}1\left(\frac{1+[v_k \succ u_k]}{3} - \Phi(\hat{\boldsymbol{z}}_n) - \boldsymbol{G}\hat{\boldsymbol{f}}\right)\Big)\end{aligned} \tag{5}$$

$$\hat{s} = \frac{2a_0 + N}{2b} \tag{6}$$

$$\hat{\ln}\, s = \Psi(2a_0 + N) - \log(2b) \tag{7}$$

$$\boldsymbol{G} = \frac{1}{2\pi}\exp\left(-\frac{1}{2}\hat{\boldsymbol{z}}_n^2\right) \tag{8}$$

where $\boldsymbol{K}_{mm}$ is the covariance between values at the inducing points, $\boldsymbol{K}_{nm}$ is the covariance between the subsample of pairwise labels and the inducing points, $\hat{\boldsymbol{z}}_n$ is the estimated preference label likelihood for the nth subsample, $b = b_0 + \frac{1}{2}\text{Tr}\left(\boldsymbol{K}_j^{-1}\left(\boldsymbol{\Sigma}_j + (\hat{\boldsymbol{f}}_j - \boldsymbol{\mu}_{j,i})(\hat{\boldsymbol{f}}_j - \boldsymbol{\mu}_{j,i})^T\right)\right)$, and Ψ is the digamma function. Given the converged estimates, we can make predictions for test arguments

with feature vectors $\boldsymbol{x}_*$. The posteriors for the latent function values $\boldsymbol{f}_*$ at the test points have mean and covariance given by:

$$\hat{\boldsymbol{f}}_* = \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m \tag{9}$$

$$\boldsymbol{C}_* = \frac{\boldsymbol{K}_{**}}{\hat{s}} + \boldsymbol{K}_{*m}\boldsymbol{K}_{mm}^{-1}(\boldsymbol{S} - \boldsymbol{K}_{mm})\boldsymbol{K}_{mm}^{-T}\boldsymbol{K}_{*m}^{T}, \tag{10}$$

where $\boldsymbol{K}_{*m}$ is the covariance between the test items and the inducing points.

3.1 Kernel Length-scale Optimsation

The prior covariance of the latent function f is defined by a kernel k, typically of the form $k_\theta(\boldsymbol{x}, \boldsymbol{x}') = \prod_{d=1}^{D} k_d(|x_d - x'_d|/l_d)$, where k_d is a function of the distance between the values of feature d for item x and x', and a length-scale hyper-parameter, l_d, which controls the smoothness of the function across the feature space. The product over features D may be replaced by other combinations, such as a sum. There are several ways to set l, including the median heuristic (Gretton et al., 2012): $l_{d,MH} = \frac{1}{D}\text{median}(\{|x_{i,d} - x_{j,d}| \forall i = 1, .., N, \forall j = 1, ..., N\})$. We can also optimise l_d by choosing the value that maximises the lower bound on the log marginal likelihood, $\mathcal{L}$, defined in Equation 3. This process is known as maximum likelihood II and is often referred to as automatic relevance determination (ARD) (Rasmussen and Williams, 2006), since features with large length-scales are less relevant because their values have less effect on $k_\theta(\boldsymbol{x}, \boldsymbol{x}')$ than features with short length-scales. The cost of optimisation may be reduced by simultaneously optimising all length-scales using a gradient-based method such as L-BFGS-B (Zhu et al., 1997). Given our proposed SVI method, we substitute our inducing point approximation into Equation 3 to approximate the gradients of $\mathcal{L}(q)$ with respect to l_d as follows:

$$\nabla_{l_d}\mathcal{L}(q) = \frac{1}{2}\hat{s}\hat{\boldsymbol{f}}_m^T\boldsymbol{K}_{mm}^{-T}\frac{\partial \boldsymbol{K}_{mm}}{\partial l_d}\boldsymbol{K}_{mm}^{-1}\hat{\boldsymbol{f}}_m - \frac{1}{2}\text{tr}\left(\left(\hat{s}\boldsymbol{K}_{mm}^{-1}\boldsymbol{S}\right)^T\left(\boldsymbol{S}^{-1} - \boldsymbol{K}_{mm}^{-1}/\hat{s}\right)\frac{\partial \boldsymbol{K}_{mm}}{\partial l_d}\right). \tag{11}$$

In our implementation, we choose the Matèrn $\frac{3}{2}$ kernel function for k due to its general properties of smoothness (Rasmussen and Williams, 2006), so that the matrix of partial derivatives is:

$$\frac{\partial \boldsymbol{K}}{\partial l_d} = \frac{\boldsymbol{K}}{k_d(|\boldsymbol{x}_d, \boldsymbol{x}'_d)}\frac{\partial \boldsymbol{K}_{l_d}}{\partial l_d}, \tag{12}$$

where each entry ij of the $\frac{\partial \boldsymbol{K}_{l_d}}{\partial l_d}$ is defined as:

$$\frac{\partial K_{d,ij}}{\partial l_d} = \frac{3|\boldsymbol{x}_{i,d} - \boldsymbol{x}_{j,d}|^2}{l_d^3}\exp\left(-\frac{\sqrt{3}|\boldsymbol{x}_{i,d} - \boldsymbol{x}_{j,d}|}{l_d}\right). \tag{13}$$

4 Experiments

We use crowdsourced preference datasets provided by Habernal and Gurevych (2016b), which contain pairwise preference labels for arguments taken from online discussion forums. Each pairwise label can have a value of 0 meaning the annotator found the second argument in the pair more convincing, 1 to express no preference, or 2 to indicate that the first argument was more convincing. All datasets contain 32 folds, which correspond to 16 controversial topics, and two stances for each topic. To test different scenarios, we apply different pre-processing steps to produce four variants of the data, which are shown in Table 1. We use the *UKPConvArgStrict* and *UKPConvArgRank* datasets to evaluate performance on 'clean' data for classification and ranking respectively. The *UKPConvArgCrowdSample* is used to evaluate both classification and ranking performance with noisy crowdsourced data including conflicts and no-preference labels.

We refer to our Bayesian preference learning method as *GPPL*. We compared our SVI method for GPPL against a less scalable variational method that did not use inducing points nor stochastic subsampling of data. Since the size of the datasets in Table 1 made it impractical to run the latter method given memory and time constraints, we tested on small subsets of the UKPConvArgStrict dataset. We found that GPPL converges to the same result using our SVI method as with variational inference. For the complete datasets listed in Table 1 we therefore report results only for our SVI approach. We compare GPPL against the *SVM* and *BLSTM* methods used

Dataset	Pairs	Arguments	No pref.	Dataset properties
UKPConvArgStrict	11642	1052	0	Combine crowdsourced labels with MACE and take $\geq 95\%$ most confident labels; Discard arguments marked as equally convincing; Discard conflicting preferences.
UKPConvArgRank	16081	1052	3289	Combine crowdsourced labels with MACE and take $\geq 95\%$ most confident labels; PageRank run on each topic to produce gold rankings. No. pairs: ; no. arguments: ; no. don't knows:
UKPConvArg-CrowdSample	16927	1052	3698	One original crowdsourced label per pair; PageRank run on each topic to produce gold rankings.

Table 1: Summary of the internet argument datasets produced using different processing steps.

in (Habernal and Gurevych, 2016b) on both pairwise classification task (predicting which of two arguments is more convincing) and a ranking task. For pairwise classifications, SVM and BLSTM concatenate the feature vectors of each pair of arguments. For ranking, PageRank scores for items in the training folds are used to train SVM and BLSTM for regression.

4.1 Experiment 1: Toy Data

Our two basic tasks are to *score* arguments in terms of convincingness and to *classify* the preference label for a pair of arguments, i.e. predict which argument is preferred. We use simulated data to show how GPPL learns differently from the pairwise labels in comparison with SVM for the classification task and PageRank for the scoring task. We simulate four scenarios, each of which contains arguments labelled *arg0* to *arg4*. In each scenario, we generate a set of pairwise preference labels. These are depicted as convincingness graphs in Figure 1a. Each scenario is repeated 25 times: in each repeat, we select arguments at random from one fold of the UKPConvArgStrict, then associate these arguments with the labels arg0 to arg4. We then obtain feature vectors for each argument by computing mean Glove word embeddings, as in Habernal and Gurevych (2016b). We trained PageRank, GPPL and the SVM classifier on the preference pairs shown in each graph and used them to predict preferences for the arguments. Taking means over the 25 repeats, we plot the PageRank scores and GPPL latent function means in Figure 1b, and the GPPL and SVM classifications for pairs of arguments in Figures 1c and 1d.

In the "no cycle" scenario, arg0 is preferred to both arg1 and arg2, which is reflected in the PageRank and GPPL scores in Figure 1b. However, arg3 and arg4 are not connected to the rest of the graph and receive different scores with PageRank and GPPL. Unlike SVM, GPPL provides probabilistic classifications and is less confident for pairs that were not yet observed, e.g. arg2 $\succ$ arg4.

The "single cycle" scenario shows how each method handles a cycle in the preference graph. Both PageRank and GPPL produce equal values for the arguments in the cycle (arg0, arg1 and arg2). PageRank assigns lower scores to both arg3 and arg4 than the arguments in the cycle, while GPPL more intuitively gives a higher score to arg3, which was preferred to arg4. SVM predicts that arg0 and arg1 are preferred over arg3, although arg0 and arg1 are in a cycle so there is no reason to prefer arg0 and arg1. GPPL, in contrast, gives a weak prediction that arg3 is preferred.

In the "double cycle" scenario, PageRank and GPPL produce very different results. Here, the argument graph shows two paths from arg2 to arg0 via arg1 or arg3, and one conflicting preference arg2 $\succ$ arg0. GPPL scores the arguments as if the single conflicting preference, arg2 $\succ$ arg0, is less impor-

tant than the two parallel paths from arg2 to arg0. In contrast, PageRank gives high scores to both arg0 and arg2. The classifications by GPPL and SVM are similar, but GPPL produces more uncertain predictions than in the first scenario due to the conflict.

Finally, "cycle with 9 undecided prefs" shows an exaggerated scenario in which we have added nine no-preference labels to the "no cycle" scenario, indicated by undirected edges, to simulate the case where multiple annotators labelled the pair and did not all agree. This does not affect the PageRank scores, but reduces the difference in GPPL scores between arg0 and the other arguments, since GPPL gives the edge from arg0 to arg0 less weight due to the undecided labels. This is reflected in the GPPL classifications, which are less confident than in the "no cycle" scenario. The SVM cannot be trained using the uncertain labels and therefore does not adapt to the undecided labels.

In conclusion, GPPL appears to resolve conflicts in the preference graphs in a more intuitive manner than PageRank, which was designed for ranking web pages by importance rather than preference. In contrast to SVM, GPPL is able to account for undecided labels to soften the latent convincingness function.

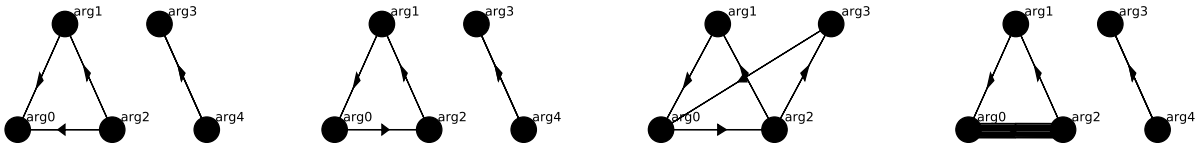
4.2 Experiment 2: Clean Data

We comparemethods for classification on the UKPConvArgStrict dataset and ranking on the UKPConvArgRank dataset. Both datasets were cleaned to remove disagreements between annotators as stated in Table 1 To choose settings for the GPPL hyperparameters a_0 and b_0, which control the noise variance, we tested three different settings and found $a_0 = 2$, $b_0 = 200$ to be most effective. This is a weak prior favouring a moderate level of noise. To set the length-scale for GPPL, we compare the median heuristic (labelled "medi.") with the MLII optimisation method (labelled as "OptGPPL"). We also compare multiplicative and additive combinations for the kernel functions for each feature. We tested GPPL with different sets of input features: 32000 linguistic features labelled as *ling*, which we also use for SVM, as in Habernal and Gurevych (2016b)); *Glove* word embeddings with 300 dimensions, which we also use for BLSTM, also as in Habernal and Gurevych (2016b); and the combination of both lin and Glove embeddings (*ling+Glove*). To create a single embedding per argument as input for GPPL, we take the mean of the individual word embeddings for the tokens in the argument.

The results are shown in Table 2. When using *ling* features, GPPL produces similar accuracy and improves the area under the ROC curve (AUC) by 2%, and cross entropy error by 0.01. Much larger improvements can be seen in the ranking metrics. When GPPL is run with mean Glove embeddings, it performs worse than BLSTM for classification but improves the ranking metrics. Using a combination of features, GPPL performs substantially better than the alternative methods for both classification and ranking, suggesting that embeddings and linguistic features contain complementary information. Optimising the length-scale using Bayesian model selection improves performance by 2% over the median heuristic. However, the cost of these improvements is that each fold required around 2 hours to compute instead of approximately 10 minutes on the same machine (an Intel i7 quad core desktop) using the median heuristic. While there is an improvement in then mean accuracy with length-scale optimisation, the accuracy does not improve for every fold. Since the optimisation step is performed using only the training folds and the model is tested on a different topic, there is a possibility of overfitting: features that are important in the test fold may not appear relevant in the training folds.

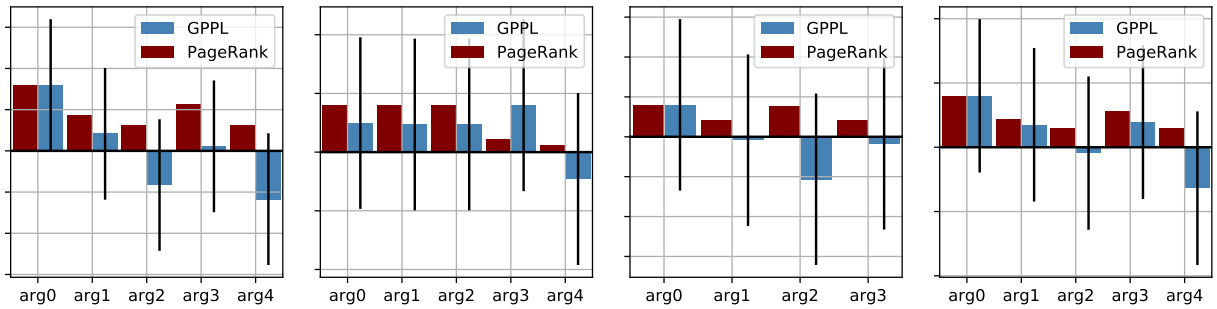
We hypothesised that GPPL benefits from integrating the GP to learn the latent preference function directly from the discrete noisy preference labels. We compare GPPL against a two stage method shown in Table 2 as *PL+SVR*: first, we use use the GPPL preference likelihood method without any item features to infer convincingness scores for each argument from the pairwise labels; second, we perform SVM regression on the inferred scores with *ling+Glove* features. The results show that PL+SVR does not reach the same performance as GPPL. This suggests that GPPL benefits not just from its preference likelihood but also from the integration of the GP.

For the pairwise classification task, we also compare GPPL against a Gaussian process classifier (*GPC*) to investigate whether other GP-based approaches produce comparable performance. As

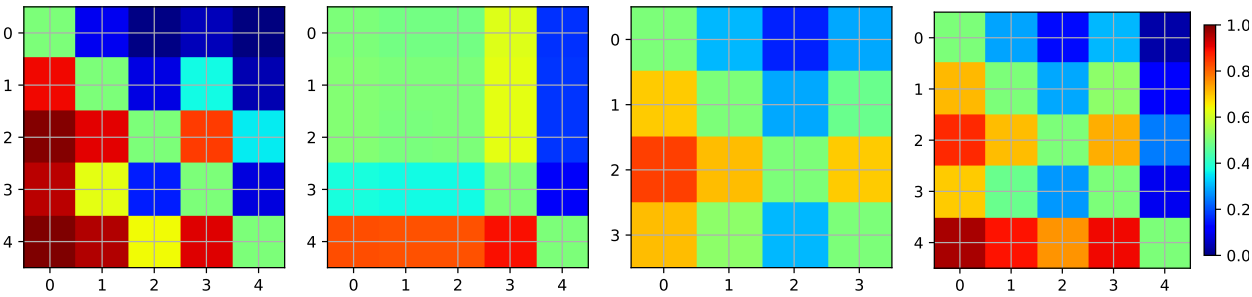


no cycle | single cycle | double cycle | cycle with 9 undecided prefs.

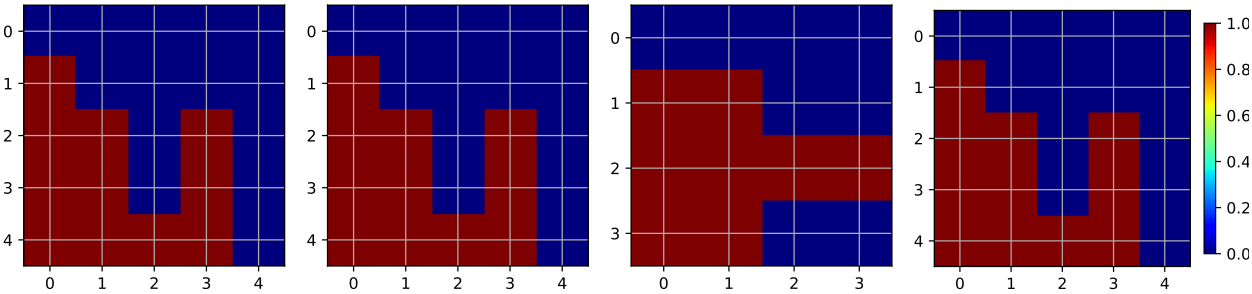
(a) Argument preference graphs for each scenario. Arrows point to the preferred argument.



(b) Ranking scores for arguments (bars for GPPL show standard deviation of latent preference function)



(c) GPPL predictions: probability that the argument on the horizontal axis is preferred to the argument on the vertical axis.



(d) SVM predictions: probability that the argument on the horizontal axis is preferred to the argument on the vertical axis.

Figure 1: Preference graphs and predictions for simulated arguments in different scenarios. The plots in each column correspond to a single scenario.

UKPConvArgStrict										
	SVM	BLSTM	GPPL*, medi.			GPPL*, opt	GPPL+, medi.	GPPL+, opt	PL +SVR	GPC
	ling	Glove	ling	Glove	ling+ Glove					
Acc.:	0.78	0.76	0.78	0.71	0.79	0.80	0.78	0.78	0.78	0.81
AUC:	0.83	0.84	0.85	0.77	0.87	0.87	0.86	0.86	0.85	0.89
CEE:	0.52	0.64	0.51	1.12	0.47	0.51	0.69	0.69	0.51	0.43
UKPConvArgRank										
Pears.:	0.36	0.32	0.38	0.33	0.45	0.44	0.40	0.40	0.39	-
Spear.:	0.47	0.37	0.62	0.44	0.65	0.67	0.64	0.64	0.63	-
Kend.:	0.34	0.27	0.47	0.31	0.49	0.50	0.49	0.49	0.47	-

Table 2: Performance comparison on clean datasets.

shown in Table 2, GPC produces the best results on the classification task, although it cannot be used to rank the arguments. While the classification approach involves learning over twice as many features – the features of the first and second items in each pair are concatenated – the GPC may perform better on this dataset because it is trained directly on the classification task, rather than through a preference learning likelihood.

4.3 Experiment 3: Conflicts and Noisy Crowdsourced Data

In this experiment, we introduced noise to both the classification and the regression tasks by comparing on the UKPConvArgCrowdSample dataset. Our goal was to investigate whether a Bayesian approach is better able to handle noise and conflicts.

The results are shown in Table 3, showing that all methods perform worse when there are noisy or conflicting preferences. GPPL and GPC produce the best results, but GPC no longer has a clear advantage over GPPL. GPPL now outperforms the other methods in all metrics except Spearman's ρ, where PL+SVR performs slightly better. It is possible that GPC and SVM have the largest changes in accuracy compared to the UKPConvArgStrict results because these classification-based methods have no mechanism to resolve conflicts in the preference graph. The performance of the BLSTM classifier also decreases by a smaller amount, but was already poorer than the other methods on UKPConvArgStrict so it is hard to compare this change directly. PL+SVR is again slightly poorer than GPPL and GPC. Metrics for ranking on UKPConvArgCrowdSample show that while GPPL and PL+SVR continue to perform well, the results for BLSTM and particularly for SVM are much poorer than with UKPConvArgRank.

UKPConvArgCrowdSample					
Acc:	0.70	0.73	0.77	0.75	0.73
AUC:	0.81	0.80	0.84	0.82	0.86
CEE:	0.58	0.54	0.50	0.55	0.53
Pears.:	0.06	0.26	0.35	0.31	-
Spear.:	0.04	0.20	0.54	0.55	-
Kend.:	0.04	0.13	0.40	0.40	-

Table 3: Performance comparison on datasets containing conflicts and noise.

4.4 Experiment 4: Active Learning

We hypothesised that a Bayesian approach would deal better with sparse data and provide more meaningful confidence estimates. To test this hypothesis, we simulated an active learning scenario, in which we simulate an agent that iteratively learns a model for each fold. Initially, $N_{inc} = 2$ pairs were chosen at random from the training set, then used to train the classifier. The agent then performs *uncertainty sampling* to select the $N_{inc} = 2$ pairs with the least confident classifications. The labels for these pairs are then taken from the training set and used to re-train the model. The result is plotted in Figure 2, showing that GPPL is able to reach accuracies above 65% with only 50 labels, while SVM and BLSTM do not reach the same performance given 200 labels. The accuracy of GPPL also increases by approximately 8% given 200 labels, while SVM increases approxi-

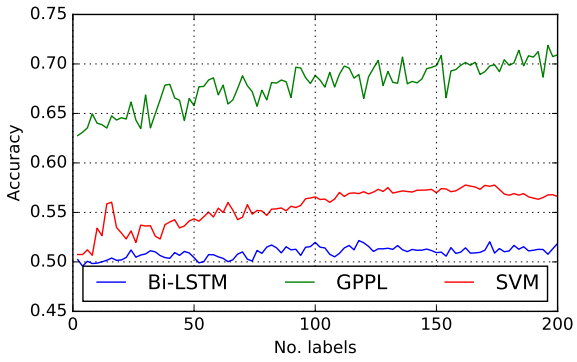


Figure 2: Active learning simulation for the three methods showing the mean accuracy of preference pair classifications over 32 runs.

mately 6% and BLSTM only 2%. This suggest that GPPL may be a more suitable model to be used with uncertainty sampling in situations where obtaining labelled data is expensive.

4.5 Experiment 5: Embeddings

In our previous experiments, we found that including mean Glove word embeddings boosted performance above only using linguistic features. However, there are several alternative methods to mean word embeddings for representing longer pieces of text, notably skip-thoughts (Kiros et al., 2015) and Siamese-CBOW (Kenter et al., 2016). We compare mean Glove embeddings with skip-thoughts embeddings and Siamese-CBOW and show the results in Table 4. The best performance was obtained using mean Glove embeddings, despite the simplicity of this approach. Further work may be required to assess whether skip-thoughts and Siamese-CBOW can be improved if trained on different corpora.

4.6 Experiment 6: Informative Features

Finally, we show how the length-scales learned by optimising GPPL can be used to identify informative sets of features. Since a larger length-scale causes greater smoothing, a very large length-scale implies that the value of that feature is irrelevant when predicting the function. In contrast, small length-scales indicate more informative features, since their precise value affects the latent preference function. Figure 3 shows the distribution of optimised length-scales on one fold of UKPConvArgStrict. The values shown are ratios of the optimised value to the

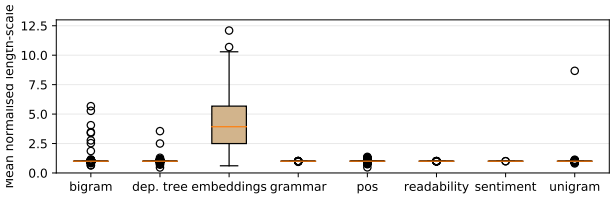


Figure 3: Distribution of length-scales for each type of feature after optimisation. Values are relative to the median heuristic value before optimisation, optimised on fold "should physical education be mandatory in schools – no", where optimisation increased accuracy from 75% to 80%.

Feature	Ratio
ProductionRule-S->ADVP,NP,VP,.,	0.466
Pos-ngram-PP-O-CARD	0.477
Unigram-"safer",	0.640
Bigram-"?"-"look"	5.672
Unigram-"safest"	8.673
Unigram-"safety"	271.190
Embedding-dimension-19	0.610
Embedding-dimension-241	12.093

Table 5: Ratios of optimised to median heuristic length-scales: largest and smallest ratios for linguistic features and word embeddings.

median heuristic. Due to the computation time required, our optimisation procedure was limited to 25 function evaluations. The large number of values close to 1 may be due to the L-BFGS-B algorithm not being able to optimise all features in the available time, suggesting that other features with larger gradients were prioritised for optimisation away from the median heuristic values. The length-scales for many dimensions of the mean word embeddings were increased, giving ratios close to 4 times the median heuristic, suggesting that these dimensions may be only very weakly informative. Table 5 shows the largest and smallest ratios for embeddings and linguistic features. The unigram "safety" has a very high length-scale, suggesting it is not informative and may be discarded.

5 Conclusions and Future Work

We presented a novel, scalable approach to predicting argument convincingness using Bayesian preference learning, and demonstrated how our method

UKPConvArgStrict									
	Median heuristic						Optimised		
	Glove	Skip-thoughts	SCBOW	ling+ Glove	ling+ Skip-th.	ling+ SCBOW	ling+ Glove	ling+ Skip-th.	ling+ SCBOW
Acc.:	0.71	0.67	0.69	0.79	0.74	0.77	0.80	0.78	0.78
AUC:	0.77	0.72	0.75	0.87	0.81	0.85	0.87	0.85	0.85
CEE:	1.12	1.11	1.22	0.47	0.80	0.52	0.51	0.51	0.50
UKPConvArgRank									
Pears.:	0.33	0.30	0.29	0.45	0.34	0.39	0.44	0.34	0.40
Spear.:	0.44	0.49	0.40	0.65	0.59	0.63	0.67	0.52	0.63
Kend.:	0.31	0.36	0.28	0.49	0.43	0.47	0.50	0.37	0.47

Table 4: Comparison between different types of embeddings with GPPL

outperforms the state-of-the-art. We showed particularly strong performance with sparse and noisy training data, as may be found in crowdsourcing or interactive learning scenarios. Future work will evaluate our approach on other NLP tasks such as the argument reasoning comprehension task (Habernal et al., 2017) where reliable classifications may be difficult to obtain. We also plan to investigate whether the GP preference function can be trained using a combination of classifications and absolute scores as well as pairwise labels.

Acknowledgments

References

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Chen Chen and Vincent Ng. 2016. Joint inference over a lightly supervised information extraction pipeline: Towards event coreference resolution for resource-scarce languages. In *AAAI*, pages 2913–2920.

Wei Chu and Zoubin Ghahramani. 2005. Preference learning with gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 137–144. ACM.

Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K Sriperumbudur. 2012. Optimal kernel choice for large-scale two-sample tests. In *Advances in neural information processing systems*, pages 1205–1213.

John Guiver and Edward Snelson. 2009. Bayesian inference for plackett-luce ranking models. In *proceedings of the 26th annual international conference on machine learning*, pages 377–384. ACM.

Ivan Habernal and Iryna Gurevych. 2016a. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *EMNLP*, pages 1214–1223.

Ivan Habernal and Iryna Gurevych. 2016b. Which argument is more convincing? Analyzing and predicting convincingness of Web arguments using bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599, Berlin, Germany. Association for Computational Linguistics.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2017. The argument reasoning comprehension task. *arXiv preprint arXiv:1708.01425*.

James Hensman, Nicolò Fusi, and Neil D Lawrence. 2013. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 282–290. AUAI Press.

James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani. 2015. Scalable Variational Gaussian Process Classification. In *AISTATS*.

Matthew D Hoffman, David M Blei, Chong Wang, and John William Paisley. 2013. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard H Hovy. 2013. Learning whom to trust with mace. In *HLT-NAACL*, pages 1120–1130.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

Maurice George Kendall. 1948. *Rank correlation methods.* Griffin.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. In *Proceedings of the The 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

David C Kingsley. 2006. Preference uncertainty, preference refinement and paired comparison choice experiments. *Dept. of Economics. University of Colorado*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Tyler Lu and Craig Boutilier. 2011. Learning mallows models with pairwise preferences. In *Proceedings of the 28th international conference on machine learning (icml-11)*, pages 145–152.

R Duncan Luce. 1959. On the possible psychophysical laws. *Psychological review*, 66(2):81.

Stephanie Lukin, Pranav Anand, Marilyn Walker, and Steve Whittaker. 2017. Argument strength is in the eye of the beholder: Audience effects in persuasion. In *15th European Chapter of the Association for Computational Linguistics (EACL)*. Association for Computational Linguistics.

David JC MacKay. 1992. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604.

Colin L Mallows. 1957. Non-null ranking models. i. *Biometrika*, 44(1/2):114–130.

Hugo Mercier and Dan Sperber. 2011. Why do humans reason? arguments for an argumentative theory. *Behavioral and brain sciences*, 34(2):57–74.

Ariel Monteserin and Analía Amandi. 2013. A reinforcement learning approach to improve the argument selection effectiveness in argumentation-based negotiation. *Expert Systems with Applications*, 40(6):2182–2188.

Frederick Mosteller. 2006. Remarks on the method of paired comparisons: I. the least squares solution assuming equal standard deviations and equal correlations. In *Selected Papers of Frederick Mosteller*, pages 157–162. Springer.

Hannes Nickisch and Carl Edward Rasmussen. 2008. Approximations for binary gaussian process classification. *Journal of Machine Learning Research*, 9(Oct):2035–2078.

Robin L Plackett. 1975. The analysis of permutations. *Applied Statistics*, pages 193–202.

Yuan Alan Qi, Thomas P Minka, Rosalind W Picard, and Zoubin Ghahramani. 2004. Predictive automatic relevance determination by expectation propagation. In *Proceedings of the twenty-first international conference on Machine learning*, page 85. ACM.

Tao Qin, Xiubo Geng, and Tie-Yan Liu. 2010. A new probabilistic model for rank aggregation. In *Advances in neural information processing systems*, pages 1948–1956.

C. E Rasmussen and C. K. I. Williams. 2006. Gaussian processes for machine learning. *The MIT Press, Cambridge, MA, USA*, 38:715–719.

Steven Reece, Stephen Roberts, David Nicholson, and Chris Lloyd. 2011. Determining intent using hard/soft data and gaussian process classifiers. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1–8. IEEE.

Ariel Rosenfeld and Sarit Kraus. 2016. Providing arguments in discussions on the basis of the prediction of human argumentative behavior. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 6(4):30.

Edwin D Simpson, Matteo Venanzi, Steven Reece, Pushmeet Kohli, John Guiver, Stephen J Roberts, and Nicholas R Jennings. 2015. Language understanding in the wild: Combining crowdsourcing and machine learning. In *Proceedings of the 24th International Conference on World Wide Web*, pages 992–1002. International World Wide Web Conferences Steering Committee.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Daniel M Steinberg and Edwin V Bonilla. 2014. Extended and unscented gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1251–1259.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th International Conference on World Wide Web*, pages 613–624. International World Wide Web Conferences Steering Committee.

Louis L Thurstone. 1927. A law of comparative judgment. *Psychological review*, 34(4):273.

Ivan Titov and Alexandre Klementiev. 2012. A bayesian approach to unsupervised semantic role induction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 12–22. Association for Computational Linguistics.

Maksims Volkovs and Richard S. Zemel. 2014. New learning methods for supervised and unsupervised preference aggregation. *Journal of Machine Learning Research*, 15(1):1135–1176.

Hui Yuan Xiong, Yoseph Barash, and Brendan J Frey. 2011. Bayesian prediction of tissue-regulated splicing using rna sequence and cellular context. *Bioinformatics*, 27(18):2554–2562.

Peng Ye and David Doermann. 2013. Combining preference and absolute judgements in a crowd-sourced setting. In *Proc. of Intl. Conf. on Machine Learning*, pages 1–7.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560.

A Variational Inference

We derive the variational lower bound as follows:

$$\mathcal{L}(q) = \sum_{i=1}^{L} \mathbb{E}_q \left[\log p\left(v_i \succ u_i | f(v_i), f(u_i)\right)\right] + \mathbb{E}_q\left[\log \frac{p\left(\boldsymbol{f}|\boldsymbol{\mu}, \boldsymbol{K}/s\right)}{q\left(\boldsymbol{f}\right)}\right] + \mathbb{E}_q\left[\log \frac{p\left(s|a_0, b_0\right)}{q\left(s\right)}\right] \tag{14}$$

Substituting the forms of the distributions with their variational parameters, we get:

$$\begin{aligned}\mathcal{L}(q) =& \mathbb{E}_q\Bigg[\sum_{i=1}^{L}[v_i \succ u_i]\log\Phi(z_i) \\ &+ [v_i \prec u_i]\left(1 - \log\Phi(z_i)\right)\Bigg] \\ &+ \log\mathcal{N}\left(\hat{\boldsymbol{f}}; \boldsymbol{\mu}, \boldsymbol{K}/\hat{s}\right) - \log\mathcal{N}\left(\hat{\boldsymbol{f}}; \hat{\boldsymbol{f}}, \boldsymbol{C}\right) \\ &+ \mathbb{E}_q\left[\log\mathcal{G}\left(s; a_0, b_0\right) - \log\mathcal{G}\left(s; a, b\right)\right]\end{aligned} \tag{15}$$

We now replace the likelihood with a Gaussian approximation:

$$\begin{aligned}\mathcal{L}(q) &\approx \mathbb{E}_q\left[\mathcal{N}(\boldsymbol{y}|\Phi(\boldsymbol{z}), \boldsymbol{Q})\right] \\ &\quad + \log\mathcal{N}\left(\boldsymbol{f}; \boldsymbol{\mu}, \boldsymbol{K}/\hat{s}\right) - \log\mathcal{N}\left(\boldsymbol{f}; \hat{\boldsymbol{f}}, \boldsymbol{C}\right) \\ &\quad + \mathbb{E}_q\left[\log\mathcal{G}\left(s; a_0, b_0\right) - \log\mathcal{G}\left(s; a, b\right)\right] \\ &\quad \approx -\frac{1}{2}\left\{L\log 2\pi + \log|\boldsymbol{Q}| - \log|\boldsymbol{C}|\right. \\ &\quad + \log|\boldsymbol{K}/s| + (\hat{\boldsymbol{f}} - \boldsymbol{\mu})\hat{s}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \\ &\quad \left.+ \mathbb{E}_q\left[(\boldsymbol{y} - \Phi(\boldsymbol{z}))^T\boldsymbol{Q}^{-1}(\boldsymbol{y} - \Phi(\boldsymbol{z}))\right]\right\} \\ &\quad - \Gamma(a_0) + a_0(\log b_0) + (a_0 - a)\mathbb{E}[\log s] \\ &\quad + \Gamma(a) + (b - b_0)\hat{s} - a\log b\end{aligned} \tag{16}$$

Finally, we use a Taylor-series linearisation to make the remaining expectation tractable:

$$\begin{aligned}\mathcal{L}(q) &\approx -\frac{1}{2}\left\{L\log 2\pi + \log|\boldsymbol{Q}| - \log|\boldsymbol{C}|\right. \\ &\quad + \log|\boldsymbol{K}/\hat{s}| + (\hat{\boldsymbol{f}} - \boldsymbol{\mu})\hat{s}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \\ &\quad \left.+ (\boldsymbol{y} - \Phi(\hat{\boldsymbol{z}}))^T\boldsymbol{Q}^{-1}(\boldsymbol{y} - \Phi(\hat{\boldsymbol{z}}))\right\} \\ &\quad - \Gamma(a_0) + a_0(\log b_0) + (a_0 - a)\mathbb{E}[\log s] \\ &\quad + \Gamma(a) + (b - b_0)\hat{s} - a\log b,\end{aligned} \tag{17}$$

where $\Gamma()$ is the gamma function, $\mathbb{E}[\log s] = \Psi(a) - \log(b)$, and $\Psi()$ is the digamma function.

The gradient of $\mathcal{L}(q)$ with respect to the length-scale, l_d, is as follows:

$$\begin{aligned}\nabla_{l_d}\mathcal{L}(q) &= -\frac{1}{2}\left\{\frac{\partial\log|\boldsymbol{K}/\hat{s}|}{\partial l_d} - \frac{\partial\log|\boldsymbol{C}|}{\partial l_d}\right. \\ &\quad \left.-(\hat{\boldsymbol{f}} - \boldsymbol{\mu})\hat{s}\frac{\partial K^{-1}}{\partial l_d}(\hat{\boldsymbol{f}} - \boldsymbol{\mu})\right\} \\ &= -\frac{1}{2}\left\{\frac{\partial\log|\frac{1}{\hat{s}}\boldsymbol{K}\boldsymbol{C}^{-1}|}{\partial l_d}\right. \\ &\quad \left.+\hat{s}(\hat{\boldsymbol{f}} - \boldsymbol{\mu})\boldsymbol{K}^{-1}\frac{\partial\boldsymbol{K}}{\partial l_d}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu})\right\}\end{aligned} \tag{18}$$

Using the fact that $\log|A| = \mathrm{tr}(\log A)$, $\boldsymbol{C} = \left[\boldsymbol{K}^{-1} - \boldsymbol{G}\boldsymbol{Q}^{-1}\boldsymbol{G}^T\right]^{-1}$, and $\boldsymbol{C} = \boldsymbol{C}^T$, we obtain:

$$\begin{aligned}&= -\frac{1}{2}\mathrm{tr}\left(\left(\hat{s}\boldsymbol{K}^{-1}\boldsymbol{C}\right)\boldsymbol{G}\boldsymbol{Q}^{-1}\boldsymbol{G}^T\frac{\partial\boldsymbol{K}}{\partial l_d}\right) \\ &\quad + \frac{1}{2}\hat{s}(\hat{\boldsymbol{f}} - \boldsymbol{\mu})\boldsymbol{K}^{-1}\frac{\partial\boldsymbol{K}}{\partial l_d}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}) \\ &= -\frac{1}{2}\mathrm{tr}\left(\left(\hat{s}\boldsymbol{K}^{-1}\boldsymbol{C}\right)\left(\boldsymbol{C}^{-1} - \boldsymbol{K}^{-1}/\hat{s}\right)\frac{\partial\boldsymbol{K}}{\partial l_d}\right) \\ &\quad + \frac{1}{2}\hat{s}(\hat{\boldsymbol{f}} - \boldsymbol{\mu})\boldsymbol{K}^{-1}\frac{\partial\boldsymbol{K}}{\partial l_d}\boldsymbol{K}^{-1}(\hat{\boldsymbol{f}} - \boldsymbol{\mu}).\end{aligned} \tag{19}$$

Assuming a product over kernels for each feature, $\boldsymbol{K} = \prod_{d=1}^{D}\boldsymbol{K}_d$, we can compute the kernel gradient as follows for the Matérn $\frac{3}{2}$ kernel function:

$$\frac{\partial\boldsymbol{K}}{\partial l_d} = \prod_{d'=1, d'\neq d}^{D} K_d\frac{\partial K_{l_d}}{\partial l_d} \tag{20}$$

$$\frac{\partial K_{l_d}}{\partial l_d} = \frac{3|\boldsymbol{x}_d - \boldsymbol{x}'_d|^2}{l_d^3}\exp\left(-\frac{\sqrt{3}|\boldsymbol{x}_d - \boldsymbol{x}'_d|}{l_d}\right) \tag{21}$$

where $|\boldsymbol{x}_d - \boldsymbol{x}'_d|$ is the distance between input points.