JMLR: Workshop and Conference Proceedings 25:1–16, 2012 Asian Conference on Machine Learning

Variational Bayesian Matching

Arto Klami ARTO.KLAMI@AALTO.FI
Helsinki Institute for Information Technology HIIT
Department of Information and Computer Science
Aalto University

Editor: Steven C.H. Hoi and Wray Buntine

Abstract

Matching of samples refers to the problem of inferring unknown co-occurrence or alignment between observations in two data sets. Given two sets of equally many samples, the task is to find for each sample a representative sample in the other set, without prior knowledge on a distance measure between the sets. Recently a few alternative solutions have been suggested, based on maximization of joint likelihood or various measures of between-data statistical dependency. In this work we present an variational Bayesian solution for the problem, learning a Bayesian canonical correlation analysis model with a permutation parameter for re-ordering the samples in one of the sets. We approximate the posterior over the permutations, and demonstrate that the resulting matching algorithm clearly outperforms all of the earlier solutions.

Keywords: Canonical correlation analysis, matching, permutation matrix, variational Bayes

1. Introduction

The task in object matching or alignment is to learn correspondence of samples in two data sets. The problem occurs frequently especially in natural language processing, in form of document alignment (Jagarlamundi et al., 2010; Quadrianto et al., 2010), sentence alignment, or word-alignment used for constructing bilingual dictionaries (Haghighi et al., 2008) and multilingual topic-models (Boyd-Graber & Blei, 2009). It also re-occurs in computational biology and medicine, where the alignment is learned between patients measured in different experiments, or between cellular level components such as protein structures (Wang & Mahadevan, 2009), functionally related genes of two species, or lipids involved in metabolism (Sysi-Aho et al., 2011; Tripathi et al., 2011). The traditional solutions for such problems create a distance measure between the two sets of objects based on domain knowledge. Given the distance, the problem is fairly straightforward to solve as an instance of bipartite graph matching. All of the works cited above, however, solve the problem in a data-driven way, by learning such a distance measure directly from the measurement data. We extend this line of work, by providing a novel Bayesian solution to the problem.

The most common assumption is that the matching is correct when some measure of statistical dependency between the two sets, measured over the objects, is maximized. Haghighi et al. (2008) and Tripathi et al. (2011) maximize correlation in a low-dimensional subspace, whereas the kernelized sorting by Quadrianto et al. (2010) maximizes the Hilbert-Schmidt Independence Criterion (HSIC; Smola et al., 2007). Haghighi et al. (2008) provides

also an alternative viewpoint: They maximize not only the correlation between the sets but also the joint likelihood of the samples under a certain probabilistic model, the probabilistic interpretation of canonical correlation analysis (CCA) (Bach and Jordan, 2005). Boyd-Graber & Blei (2009) give another example of learning the match to maximize the joint likelihood, and the early kernelized sorting algorithm by Jebara (2004) maximizes the joint likelihood of a Gaussian model.

We extend the work of Haghighi et al. (2008) in the sense that we define the matching through both a joint likelihood criterion and maximal correlation. The former criterion is a natural choice in the generative modeling framework, whereas the latter choice is encouraged by the success of earlier dependency-maximizing solutions. The main difference between the two solutions is that Haghighi et al. (2008) learns the maximum likelihood estimate over the match and the parameters of the probabilistic model, whereas we do posterior inference for both. As the underlying model we use the Bayesian canonical correlation analysis (Wang, 2007; Klami & Kaski, 2007; Virtanen et al., 2011).

The match can be parameterized by a permutation matrix. If $\boldsymbol{X}$ denotes one data set of N objects and $\boldsymbol{Y}$ the other, the task is to find a $N \times N$ permutation matrix $\boldsymbol{\pi}$ such that $\boldsymbol{X}$ is close to $\boldsymbol{Y}\boldsymbol{\pi}$, where "close" is defined by one of the criteria mentioned above. To be more precise, we will do inference over a posterior distribution of the permutation. Exact variational inference over permutations is only tractable for small N (at most tens), based on Fourier transformations (Kondor et al., 2007) or mapping the permutations to a hypersphere (Plis et al., 2011). To work with larger N, we present two alternative approximative techniques that are computationally feasible for the size of problems considered by earlier matching solutions (hundreds of samples). Both build on the observation that we can learn the most likely permutation efficiently with an assignment problem (AP) solver.

In summary, we present a novel solution to the matching problem using variational Bayesian inference. We simultaneously learn the posterior over parameters of a Bayesian CCA model and a permutation matrix representing the match. We compare the method with the best earlier solutions on two benchmark datasets used by Jagarlamundi et al. (2010), Quadrianto et al. (2010) and Tripathi et al. (2011), and show how the new variational Bayesian matching clearly outperforms them. Furthermore, we demonstrate the importance of good initialization and provide a new initialization strategy that is generally applicable for all matching solutions.

2. Matching

2.1. Problem Formulation

Given two sets of N objects, denoted by $\mathcal{X}$ and $\mathcal{Y}$, the goal is to discover a permutation matrix $\boldsymbol{\pi}$ over the objects in $\mathcal{Y}$ such that the ith object in $\mathcal{X}$ corresponds to the object in $\mathcal{Y}$ for which $\boldsymbol{\pi}_{ji} = 1$. The correspondence is defined by a cost function $c(\mathcal{X}, \mathcal{Y}|\boldsymbol{\pi})$ which is maximized with respect to $\boldsymbol{\pi} \in \mathcal{P}$. Here $\mathcal{P} \in \{0,1\}^{N \times N}$ is the set of all $N \times N$ binary permutation matrices with unit row and column sums. Our cost function will be the marginal log-likelihood, whereas many of the earlier solutions have used statistical dependency.

In this work the samples will be represented as real-valued vectors, and hence the sets are represented as matrices $\boldsymbol{X} \in \mathbb{R}^{D_x \times N}$ and $\boldsymbol{Y} \in \mathbb{R}^{D_y \times N}$. Individual samples will be denoted by column vectors $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$.

2.2. Earlier Solutions

Most earlier solutions to the matching problem maximize a dependency measure such as canonical correlation or the HSIC measure Smola et al. (2007). Haghighi et al. (2008) and Tripathi et al. (2011) seek for a linear subspace revealing the correlations, alternatingly optimizing over the projections and solving a linear assignment problem to find the best permutation in that subspace. The kernelized sorting methods Jebara (2004); Quadrianto et al. (2010), in turn, directly evaluate and maximize a kernel-based dependency measure. However, the resulting cost corresponds to a quadratic assignment problem, which they solve by iteratively applying a linear AP solver. Jagarlamundi et al. (2010) presented improved initialization and various other tricks to improve robustness of kernelized sorting solutions.

The matching canonical correlation analysis (MCCA) method by Haghighi et al. (2008) can also be interpreted as maximizing the joint likelihood of the two data sets, based on the probabilistic interpretation of CCA (Bach and Jordan, 2005). Another example of likelihood maximization is the multilingual topic model of Boyd-Graber & Blei (2009), which matches vocabularies of two languages by finding a maximum a posteriori estimate over their permutations. Finally, an alternative strategy for learning the match is to explicitly create a distance measure between the two sets. For example, manifold alignment by Wang & Mahadevan (2009) does that by comparing within-set neighborhoods. Given the distances, the matching problem can be solved as linear AP. While this approach circumvents the need for iterative solution, the computational cost for learning the distances is severe.

We will compare our method with the closest alternatives from all categories, namely the alternating CCA+AP solution by Tripathi et al. (2011), the leading kernelized sorting variant by Jagarlamundi et al. (2010), and the manifold alignment method of Wang & Mahadevan (2009) complemented with an AP solver. The model by Boyd-Graber & Blei (2009) is not comparable, since their goal is learning a topic-model and the match is learned only as a part of it.

3. Variational Bayesian Matching

We formulate a solution to the matching problem by straightforward joint modeling. We assume the data is generated by a latent variable model of the type

$$p(\boldsymbol{X}, \boldsymbol{Y}) = \prod_{i=1}^{N} \int p(\boldsymbol{x}_i|\boldsymbol{z}_i) p(\boldsymbol{y}_i|\boldsymbol{z}_i) p(\boldsymbol{z}_i) d\boldsymbol{z}_i.$$

That is, the i.i.d. samples in the two data sets are conditionally independent given a latent variable, which is a typical assumption in multi-view learning.

The matching is introduced as an explicit permutation matrix $\boldsymbol{\pi} \in \mathcal{P}$ applied to re-order the samples in one of the data sets. We indicate by $\boldsymbol{\pi}_{ji} = 1$ that the sample $\boldsymbol{y}_j$ pairs with the latent variable $\boldsymbol{z}_i$ (which is associated with the sample $\boldsymbol{x}_i$), and using $\boldsymbol{\pi}_{\cdot i}$ to denote the

ith column we can write the Bayesian matching model as

$$p(\boldsymbol{X}, \boldsymbol{Y}) = \int_{\mathcal{P}} \left(p(\boldsymbol{\pi}) \prod_{i=1}^{N} \int p(\boldsymbol{x}_i|\boldsymbol{z}_i) p(\boldsymbol{Y}\boldsymbol{\pi}_{\cdot i}|\boldsymbol{z}_i) p(\boldsymbol{z}_i) d\boldsymbol{z}_i \right) d\boldsymbol{\pi}.$$

We will explicitly integrate over $\boldsymbol{\pi}$, which turns out to be crucially important for the matching accuracy. To compare with the alternative methods only providing a point-estimate, we will additionally obtain a MAP estimate for $\boldsymbol{\pi}$ after learning the posterior.

We could alternatively introduce separate permutation matrices for both $\boldsymbol{X}$ and $\boldsymbol{Y}$. This would make the scenario more symmetric by enabling soft choice of latent variables for both sets, but for the task of learning a match it would mostly just double the computational cost due to needing to optimize for two permutations.

3.1. Bayesian CCA

As the actual model we use the Bayesian CCA model as presented by Virtanen et al. (2011), which matches the choice of maximizing correlation made by the earlier solutions of Haghighi et al. (2008) and Tripathi et al. (2011). This means our approach is maximizing both the joint marginal likelihood and a dependency measure. Furthermore, it has the intuitively appealing property that we need not consider variation independent of the other set while learning the match.

The Bayesian CCA with K components is defined as

$$\begin{aligned} \boldsymbol{z}_i &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \\ [\boldsymbol{x}_i; \boldsymbol{y}_i] &\sim \mathcal{N}(\boldsymbol{W}\boldsymbol{z}_i, \boldsymbol{\Sigma}), \end{aligned}$$

where $[\boldsymbol{x}_i; \boldsymbol{y}_i]$ denotes the feature-wise concatenation of the samples with $D = D_x + D_y$ dimensions and $\boldsymbol{z}_i \in \mathbb{R}^{K\times 1}$. The covariance is diagonal

$$\boldsymbol{\Sigma} = \begin{bmatrix} \tau_x^{-1}\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \tau_y^{-1}\boldsymbol{I} \end{bmatrix},$$

with Gamma prior for the precisions τ_x and τ_y, and the projection $\boldsymbol{W} \in \mathbb{R}^{D\times K}$ is made group-wise sparse by the prior

$$\begin{aligned} \beta_{xk} &\sim \mathcal{G}(\alpha_0, \beta_0) \qquad \beta_{yk} \sim \mathcal{G}(\alpha_0, \beta_0) \\ p(\boldsymbol{W}) &= \prod_{k=1}^{K} \left(\mathcal{N}(\boldsymbol{W}_x(k)|\boldsymbol{0}, \beta_{xk}^{-1}\boldsymbol{I}) \mathcal{N}(\boldsymbol{W}_y(k)|\boldsymbol{0}, \beta_{yk}^{-1}\boldsymbol{I}) \right). \end{aligned}$$

Here $\boldsymbol{W} = [\boldsymbol{W}_x; \boldsymbol{W}_y]$ so that $\boldsymbol{W}_x$ covers the dimensions spanned by $\boldsymbol{X}$, $\boldsymbol{W}_x(k)$ is the kth column of $\boldsymbol{W}_x$, and $\mathcal{G}(\alpha_0, \beta_0)$ is a flat Gamma prior. The prior is critical in dividing the components into three categories: those describing correlations between the two sets (β_{xk} and β_{yk} are both small) and to those describing variation specific to just one set and independent of the other (either β_{xk} or β_{yk} is very large).

An efficient variational approximation is provided by the factorization

$$Q = q(\tau_x)q(\tau_y) \prod_{k=1}^{K} q(\beta_{xk})q(\beta_{yk}) \prod_{i=1}^{N} q(\boldsymbol{z}_i) \prod_{d=1}^{D} q(\boldsymbol{W}_{d\cdot}), \tag{1}$$

and the parameters of each term are learned with the alternating updates explained by Virtanen et al. (2011).

3.2. Matching Bayesian CCA

The matching Bayesian CCA model extends the above formulation by replacing the likelihood part with

$$[\boldsymbol{x}_i; \boldsymbol{Y}\boldsymbol{\pi}_{.i}] \sim \mathcal{N}(\boldsymbol{W}\boldsymbol{z}_i, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\pi} \in \mathcal{P}$. We assume a uniform prior over all $N \times N$ permutation matrices, but could easily incorporate priors obtained from additional information sources as long as they factorize over the sample pairs.

The variational approximation in (1) needs to be complemented by an extra term $q(\boldsymbol{\pi})$, which is a distribution over permutation matrices. The primary challenge in learning the model is in updating this distribution. Even approximating the distribution that is over the $N!$-dimensional space initially sounds infeasible, but in practice the problem is simplified by two observations: (i) We can efficiently find the exact permutation matrix $\hat{\boldsymbol{\pi}}$ that maximizes the variational lower bound and (ii) only a tiny fraction of other permutation matrices will have probability that does not vanish to zero compared to $\hat{\boldsymbol{\pi}}$ (see, e.g., Leskovec et al. (2010)). Given these observations, we can construct $q(\boldsymbol{\pi})$ by properly normalizing a distribution over a small set of permutations $S = \{\boldsymbol{\pi}^{(m)}\}_{m=0}^{M}$ that includes $\hat{\boldsymbol{\pi}} = \boldsymbol{\pi}_0$ and some nearby permutations.

Given a set S of feasible permutation matrices with their associated weights w_m, it is easy to compute the expectation $\langle\boldsymbol{\pi}\rangle = \frac{1}{\sum w_m}\sum_{m=0}^{M} w_m \boldsymbol{\pi}^{(m)}$. Simple verification confirms that the update rules of regular Bayesian CCA can be re-used for all other terms in the approximation, assuming we simply transform $\boldsymbol{Y}$ by $\langle\boldsymbol{\pi}\rangle$ after updating the match.

3.2.1. Finding the Most Likely Permutation

The log-cost of any permutation is given by

$$\log w_m \propto -\frac{1}{2}\sum_{i=1}^{N}\left\langle \tau_y \left(\boldsymbol{Y}\boldsymbol{\pi}_{.i}^{(m)} - \boldsymbol{W}_y\boldsymbol{z}_i)\right)^T \left(\boldsymbol{Y}\boldsymbol{\pi}_{.i}^{(m)} - \boldsymbol{W}_y\boldsymbol{z}_i)\right)\right\rangle,$$

where $\langle\cdot\rangle$ denotes the expectation over the approximating posterior. To find the most likely permutation $\hat{\boldsymbol{\pi}}$, we collect all pairwise expected distances into a single $N \times N$ matrix $\boldsymbol{A}$ with entries

$$\begin{aligned}\boldsymbol{A}_{ij} &= \frac{1}{2}\langle\tau_y(\boldsymbol{y}_j - \boldsymbol{W}_y\boldsymbol{z}_i)^T(\boldsymbol{y}_j - \boldsymbol{W}_y\boldsymbol{z}_i)\rangle \\ &= \frac{1}{2}\langle\tau_y\rangle\left[\boldsymbol{y}_j^T\boldsymbol{y}_j - 2\boldsymbol{y}_j^T\langle\boldsymbol{W}_y\rangle\langle\boldsymbol{z}_i\rangle + \mathrm{Tr}\left(\langle\boldsymbol{W}_y^T\boldsymbol{W}_y\rangle\langle\boldsymbol{z}_i\boldsymbol{z}_i^T\rangle\right)\right], \end{aligned} \tag{2}$$

where $\mathrm{Tr}(\cdot)$ denotes the matrix trace. A regular linear assignment problem solver will then find a globally optimal choice of $\hat{\boldsymbol{\pi}}$ so that $\sum\mathrm{diag}(\boldsymbol{A}\hat{\boldsymbol{\pi}})$ is minimized.

It is worth noting that the entries of $\boldsymbol{A}_{ij}$ are here assumed independent; the uncertainty in $\boldsymbol{z}_i$ is integrated out separately for each j. For any single $\boldsymbol{\pi}$ this corresponds directly to the factorization assumptions of (1) and hence we will correctly identify the most likely

permutation without needing to make any further assumptions. However, for comparing different permutations the above formulation makes one more independence assumption: The latent variables are assumed independent of the match. This is a necessary assumption for efficient computation, but as discussed in Section 3.2.2 it causes problems when trying to approximate the full posterior over the matches, especially for high-dimensional data. We will later explain one way of relaxing the assumption, but first we proceed by explaining the rest of the algorithm for this factorized choice.

Another important detail concerns the computation of the expectations in (2). The straightforward choice would be to use as $\langle \boldsymbol{z}_i \rangle$ and $\langle \boldsymbol{z}_i \boldsymbol{z}_i^T \rangle$ the moments of the current $q(\boldsymbol{z}_i)$, which has been computed according to the previously updated $q(\boldsymbol{\pi})$. This would, however, bias the choice of the permutation towards the current expectation $\langle \boldsymbol{\pi} \rangle$, preventing efficient change of pairs and causing severe issues with local optima: if $\boldsymbol{y}_j$ was last time paired with $\boldsymbol{x}_i$ then $\boldsymbol{z}_i$ has already been optimized for $\boldsymbol{y}_j$. A much more efficient approach is to implicitly re-optimize $q(\boldsymbol{z}_i)$ for each entry of $\boldsymbol{A}$, allowing fair game between the different samples. Assuming $\boldsymbol{y}_j$ pairs with $\boldsymbol{z}_i$, the update rule for $q(\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{\mu}_{ij}, \boldsymbol{\Sigma}_z)$ is given by

$$\begin{aligned} \boldsymbol{\Sigma}_z &= \left(\langle \tau_x \rangle \langle \boldsymbol{W}_x^T \boldsymbol{W}_x \rangle + \langle \tau_y \rangle \langle \boldsymbol{W}_y^T \boldsymbol{W}_y \rangle + \boldsymbol{I} \right)^{-1} \\ \boldsymbol{\mu}_{ij} &= \boldsymbol{\Sigma}_z \left(\langle \tau_x \rangle \langle \boldsymbol{W}_x^T \rangle \boldsymbol{x}_i + \langle \tau_y \rangle \langle \boldsymbol{W}_y^T \rangle \boldsymbol{y}_j \right). \end{aligned} \tag{3}$$

Plugging $\langle \boldsymbol{z}_i \rangle = \boldsymbol{\mu}_{ij}$ and $\langle \boldsymbol{z}_i \boldsymbol{z}_i^T \rangle = \boldsymbol{\mu}_{ij} \boldsymbol{\mu}_{ij}^T + \boldsymbol{\Sigma}_z$ in (2) improves the eventual matching accuracy of the method considerably, compared to directly using the current $q(\boldsymbol{z}_i)$. Note that $\boldsymbol{\Sigma}_z$ does not depend on i or j, and hence needs to be computed only once per iteration.

Finally, since CCA automatically infers which of the K components model correlations between the two sets, it makes sense to learn the match only over those components. We can do that by analytically integrating out $\boldsymbol{z}_i$ for the components that do not describe any variation in $\boldsymbol{X}$ (that is, β_{xk} is very large) – the ones not active in $\boldsymbol{Y}$ anyway have no effect on $\boldsymbol{A}$. Denoting by $\boldsymbol{V}_y$ the columns of $\boldsymbol{W}_y$ corresponding to the components marginalized out and by $\boldsymbol{U}_y$ the remaining columns, the entries in (2) are replaced by $\frac{1}{2} \langle (\boldsymbol{y}_j - \boldsymbol{U}_y \boldsymbol{z}_i) \boldsymbol{\Psi} (\boldsymbol{y}_j - \boldsymbol{U}_y \boldsymbol{z}_i) \rangle$, where $\boldsymbol{\Psi} = \left(\boldsymbol{V}_y \boldsymbol{V}_y^T + \tau_y^{-1} \boldsymbol{I} \right)^{-1}$. For efficient inversion we use the Woodbury identity $\boldsymbol{\Psi} = \tau_y \boldsymbol{I} - \tau_y^2 \boldsymbol{V}_y \left(\boldsymbol{I} + \tau_y \boldsymbol{V}_y^T \boldsymbol{V}_y \right)^{-1} \boldsymbol{V}_y^T$. For large D_y we can further approximate $\boldsymbol{\Psi} = \tau_y \boldsymbol{I}$ for faster computation, since the elements of the latter term are of the order $\tau_y \frac{\mathrm{rank}(\boldsymbol{V}_y)}{D_y}$, tending to zero for large D_y. This can be seen (details omitted for brevity) by contrasting the scales of $\boldsymbol{V}_y \boldsymbol{V}_y^T$ and $\boldsymbol{V}_y^T \boldsymbol{V}_y$, for which we get analytic expressions since the elements of $\boldsymbol{V}_y$ are drawn from zero-mean Gaussian distributions with known variances.

3.2.2. FINDING OTHER LIKELY PERMUTATIONS

To create a full distribution over the permutations we need to complement the most likely permutation with a set of other reasonable permutations. For this purpose we propose two alternative strategies. The first approach assumes the latent variables to be independent of the match and results in a deterministic approximation, whereas the latter breaks the independent assumption but numerical integration is needed for inference.

Factorized Method: By making the assumption that $\boldsymbol{z}_i$ are independent of the permutation $\boldsymbol{\pi}$, we can directly use costs of the form $w_m \propto e^{-\sum \mathrm{diag}(\boldsymbol{A}\boldsymbol{\pi}^{(m)})}$ to obtain the relative

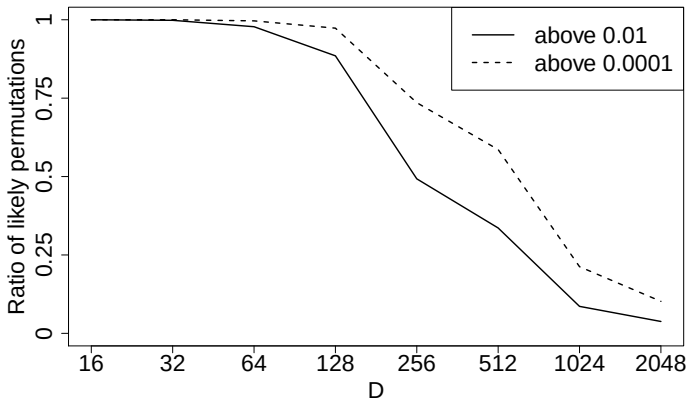

Figure 1: The ratio of likely permutations (probability compared to the best permutation $\hat{\boldsymbol{\pi}}$ above 10^{-2} or 10^{-4}) in the neighborhood of $\hat{\boldsymbol{\pi}}$ as a function of the dimensionality D. For large D all of the alternatives have essentially zero probability, which implies that $q(\boldsymbol{\pi})$ converges towards a delta distribution around $\hat{\boldsymbol{\pi}}$. The ratios were computed for the final model learned for $N = 80$ images (see Section 4.1) based on random subsets of D features, averaged over 10 runs. For larger N the situation is not as extreme, but the trend is similar.

probabilities of different permutations $\boldsymbol{\pi}^{(m)}$. We can then approximate the full posterior by repeatedly slightly perturbing $\hat{\boldsymbol{\pi}}$ and computing the relative cost of the resulting alternative permutations. This will produce a unimodal approximation centered around the mode, to complement the other unimodal terms in (1).

To create the set of other feasible permutations, we find N other permutations that differ minimally from the optimal one. We exclude one pair at a time from the optimal match (by setting the corresponding element in $\boldsymbol{A}$ to infinity) and solve the AP again. All of the resulting matches $\boldsymbol{\pi}^{(m)}$ will be worse than the optimal one, but will be maximally close in terms of probability due to having only one extra constraint. Note, however, that they will typically differ for multiple pairs, since the single constraint propagates to multiple changes in the full permutation. For each unique $\boldsymbol{\pi}^{(m)}$ we then evaluate the associated cost w_m, and the expectation over the approximative posterior is given by the weighted average $\langle \boldsymbol{\pi} \rangle = \frac{1}{\sum w_m} \sum_{m=0}^{M} w_m \boldsymbol{\pi}^{(m)}$. Here $\boldsymbol{\pi}^{(0)} = \hat{\boldsymbol{\pi}}$ denotes the most likely permutation. Note that $M \leq N$, since we create N alternative permutations, but it is possible that several constraints result in the same permutation.

An interesting observation is that for large D_y the weight w_m will be negligible for most $\boldsymbol{\pi}^{(m)}$. Even though the algorithm generates maximally similar permutations by adding always just one additional constraint, most of them have very low probability for all but very low-dimensional data. We ran the algorithm for various lower-dimensional variants of the image data studied in Section 4.1, and plot the ratio of likely permutations as a function of dimensionality in Figure 1. Already for $D_y = 128$ the relative probabilities of half of the permutations are below 10^{-4}, and for $D_y = 2048$ we find essentially no permutations that would have noticeable weight in $q(\boldsymbol{\pi})$.

For low-dimensional data the posterior over the permutations is smooth, and no efficient algorithm for finding all likely permutations exists. The above process will, however, generate a reasonable subset of those in the immediate vicinity of the optimal one, providing an approximation of the posterior around its mode. It is also easy to extend the procedure to create larger set of likely permutations, for example by creating also alternative permutations with more than one constraint, or by performing weighted random walk in the space of all permutations similar to the sampling scheme by Leskovec et al. (2010). In the experiments we use the algorithm as described above, creating only the N alternative permutations. Preliminary tests indicate that covering bigger set of likely permutations slightly improves the accuracy, but for these applications the gain is small compared to the increased computational cost.

Non-factorized Method: The property that $q(\boldsymbol{\pi})$ converges to a delta distribution around $\hat{\boldsymbol{\pi}}$ is caused by the assumption that the latent variables are independent of the match. When computing $\boldsymbol{A}_{ij}$ we integrate over $\boldsymbol{z}_i$ separately for each j, which implies that the matching cost depends only on the expected distances between $\boldsymbol{W}_y\boldsymbol{z}_i$ and $\boldsymbol{y}_j$. This assumption is critical for computation (otherwise the matching cost would not factorize over the samples), but it also means that the uncertainty in the distances is not fully taken into account: the pairs with the smallest expected distance will be strongly preferred for high-dimensional data, irrespective of the variance.

To avoid the problem we should model the dependencies between the choices instead of allowing each $\boldsymbol{y}_j$ to independently integrate over $\boldsymbol{z}_i$. If a particular value is chosen for $\boldsymbol{z}_i$ then it should simultaneously become, for instance, both a better pair for $\boldsymbol{y}_j$ and a worse pair for some other sample $\boldsymbol{y}_l$. In other words, the posterior distributions depend on the chosen match: Conditional on the choice "$\boldsymbol{y}_j$ is paired with $\boldsymbol{x}_i$", the distribution of $\boldsymbol{W}_y\boldsymbol{z}_i$ shifts closer to $\boldsymbol{y}_j$ than what the update rule in (3) would suggest, and often this implies it shifts away from some other samples. This property is illustrated in Figure 2.

Explicitly modeling such dependencies requires simultaneously integrating over the N latent variables $\boldsymbol{z}_i$ to compute the relative costs of different permutations. Solving such an integral analytically seems hopeless, and hence we resort to Monte Carlo integration. For notational convenience, we introduce a new variable $\boldsymbol{\xi}_i \sim \mathcal{N}(0, \boldsymbol{\Sigma}_z)$ such that $\boldsymbol{z}_i = \boldsymbol{\mu}_{ij} + \boldsymbol{\xi}_i$ where $\boldsymbol{\mu}_{ij}$ and $\boldsymbol{\Sigma}_z$ are given in (3). Instead of assuming independent set of $\boldsymbol{\xi}_i$ values for each sample $\boldsymbol{y}_j$ and integrating them out (which would correspond to (2)), we draw a joint random sample $\{\boldsymbol{\xi}_i^{(m)}\}_{i=1}^N$ that is independent of j. This results in $\boldsymbol{z}_i^{(m)}|\boldsymbol{x}_i, \boldsymbol{y}_j = \boldsymbol{\mu}_{ij} + \boldsymbol{\xi}_i^{(m)}$ for all choices of $\boldsymbol{y}_j$. We then solve the assignment problem with costs $\boldsymbol{A}_{ij} = \frac{1}{2}\langle\tau_y(\boldsymbol{y}_j - \boldsymbol{W}_y\boldsymbol{z}_i^{(m)})^T(\boldsymbol{y}_j - \boldsymbol{W}_y\boldsymbol{z}_i^{(m)})\rangle$, where $\boldsymbol{z}_i^{(m)}$ is now a fixed quantity and hence the expectation is only over $q(\tau_y)$ and $q(\boldsymbol{W})$, to find the match corresponding to this particular choice of $\boldsymbol{\xi}^{(m)}$. By repeating this process M times we can estimate $\langle\boldsymbol{\pi}\rangle$ as the unweighted average of the resulting permutations.

It is worth noting that the above numerical integration scheme does not result in monotonically increasing variational lower bound for the marginal likelihood. We have not found this to be a problem in practice, and as shown by the experiments the resulting match is better than the one obtained when not modeling the dependencies between the latent variables. The gain is particularly clear for high-dimensional data.

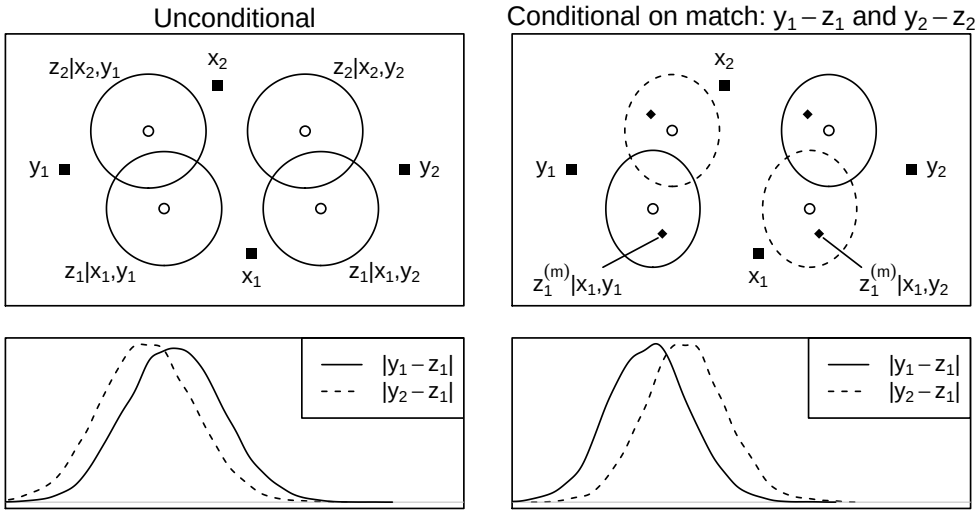

Figure 2: Schematic illustration of the matching process, drawn for $\boldsymbol{W}_x = \boldsymbol{W}_y = \boldsymbol{I}$ which enables depicting both the latent variables and the samples of both views in the same two-dimensional space. **Left:** The posterior density of $\boldsymbol{z}_i$ depends on both $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$, and for two samples in both views there are a total of four peaks in the posterior of the latent variables, one for each possible pair. The factorized variational approximation integrates over the uncertainty (denoted by the equiprobability contours) for each choice independently, and the match will only depend on the difference of the expected distances. Even though the distributions of the distances to $\boldsymbol{z}_1$ (bottom figure) overlap heavily, the closer sample will be preferred strongly for high-dimensional data. **Right:** Conditional on having made a particular match (here $\boldsymbol{y}_1$ pairing with $\boldsymbol{z}_1$ and $\boldsymbol{y}_2$ pairing with $\boldsymbol{z}_2$) the posterior distributions of $\boldsymbol{z}_i$ shift towards the chosen $\boldsymbol{y}$ samples, influencing not only the distances depicted here but also distances to potential other samples $\boldsymbol{y}_l$. If this effect is not modeled, the permutations with some unlikely pairs will receive too low probability since the compensation from these posterior shifts is ignored. We model the dependencies by sampling: For each $\boldsymbol{z}_i$ we draw $\boldsymbol{\xi}_i^{(m)}$ that is converted to $\boldsymbol{z}_i^{(m)}|\boldsymbol{x}_i, \boldsymbol{y}_j$ (see text for details) shown here as small diamonds, and then solve the match using distances from $\boldsymbol{y}_j$ to these samples.

3.2.3. Initialization

As demonstrated by most earlier works, the matching problem is very sensitive to the initialization. Our solution is no exception, due to the iterative mean-field algorithm for updating the variational approximation. Hence, we present an initialization scheme that borrows elements from several earlier solutions.

The basic idea is that we solve the problem L times, each time with a different initialization. We then create a consensus of those matches, following the idea by Tripathi et al. (2011), by counting how many times each of the sample pairs were matched together

in the set of the L solutions. Finally, the actual solution is computed by initializing the model with the consensus (normalized to probabilities) and solving the matching problem one more time.

The reasoning behind the strategy is that by choosing diverse initial models we can better cover the space of potential matches, making the unimodal posterior approximation less of a limitation. However, each individual initialization should still be a good one. To achieve both properties, we use a modified version of the PCA-initialization suggested by Quadrianto et al. (2010). For each of the L initializations we compute PCA separately for both sets and order the samples according to the first component. For increasing the diversity, we compute the PCA from random subset of $N/2$ dimensions instead of the whole data, getting slightly different initialization for each run. Furthermore, we smoothen the initial permutation by convolving it with a Gaussian kernel (the exact width does not seem to matter).

3.3. Summary of the Method

Due to the somewhat complicated presentation above, we here summarize the alternative versions of the algorithm. All choices are based on the same idea of learning Bayesian CCA with an extra permutation matrix for re-ordering the samples, and they optimize each term in the posterior approximation alternatingly. For all other parameters we re-use the updates by Virtanen et al. (2011) but replace $\boldsymbol{Y}$ with $\boldsymbol{Y}\langle\boldsymbol{\pi}\rangle$. For updating $q(\boldsymbol{\pi})$ we have three alternative choices which will be empirically compared in the next section:

1. **Hard matching (Hard)**: We find the most likely permutation $\hat{\boldsymbol{\pi}}$ and assume a delta distribution for the posterior, such that $\langle\boldsymbol{\pi}\rangle = \hat{\boldsymbol{\pi}}$.

2. **Factorized soft matching (F-Soft)**: The most likely permutation is complemented with a set of other feasible permutations $\{\boldsymbol{\pi}^{(m)}\}$, obtained by re-optimizing the match with extra constraints that prevent each of the samples in turn from picking its favorite pair. We then compute $\langle\boldsymbol{\pi}\rangle$ as a weighted average of these permutations.

3. **Non-factorized soft matching (NF-Soft)**: We numerically integrate over $q(\boldsymbol{z}_i)$ to model the dependencies between the match and the latent variables. For each $\boldsymbol{Z}^{(m)}$ drawn from the posterior we solve the assignment problem to obtain a feasible permutation $\boldsymbol{\pi}^{(m)}$. The expectation is given by the flat average of such permutations.

For all variants we learn the match only over the components for which both β_{xk} and β_{yk} are small (in practice, we use components that describe at least 0.1% of the variance in both data sets), to ignore the set-specific variation not useful for learning the match. Learning the match is the most time-consuming part, and hence we update $q(\boldsymbol{\pi})$ only after every 10 iterations for the other terms. Overall, the computational cost of the method is dominated by the cost used for computing the $\boldsymbol{A}$ matrix and solving the assignment problem. The hard variant does these only once per iteration, the F-Soft variant repeats the latter N times, and the NF-Soft variant repeats both multiple (in our experiments $M = 50$) times for each iteration.

4. Experiments

Next we compare the alternative versions presented above with earlier matching problem solutions. For fair comparison, we use data sets analyzed by the authors of the earlier methods. We start with the image data used by Quadrianto et al. (2010) and Jagarlamundi et al. (2010) and compare our method with the latest kernelized sorting variant, and then proceed to compare our method with a CCA-based method and a distance-based method of Wang & Mahadevan (2009) on the data used by Tripathi et al. (2011). The first data is high-dimensional ($D_x = D_y = 2400$), and hence an example of a case where the factorized soft matching does not help, whereas the latter is low-dimensional (the dimensionalities range from 3 to 30) and both of the soft variants provide a proper distribution over the permutations.

4.1. Image Matching

In this problem the task is to match two halves of a set of 320 images, using the raw pixels values (40×40 pixels in Lab color space) as the input. The problem itself is completely artificial, but it has nevertheless become a kind of benchmark for the matching solutions due to the data provided by Quadrianto et al. (2010).

We apply the hard and non-factorized soft variants of our model to different subsets of the data. For each choice of N, we learn $L = 50$ initial solutions and initialize the final model by the consensus of these matches. To further diversify the initializations, we use only a subset of $300 - 1200$ dimensions (random choice) for each initial run. We use $K = 8$ components, but the results would not change much for larger K since the Bayesian CCA automatically ignores unnecessary components ($K = 16$ gave slightly more correct matches, 216, for the full data, but slightly less for some smaller values of N).

Figure 3 compares the proposed method with the p-smooth variant of kernelized sorting by Jagarlamundi et al. (2010), which has earlier been demonstrated to be superior to the original algorithm. The p-smooth model is slightly better than ours for small N, but starting from roughly $N = 100$ the NF-Soft variant clearly outperforms it. In particular, for the choice of all images we get almost 50% more correct matches (201 vs 136).[1] The figure also shows that it pays off to use a full distribution over $\boldsymbol{\pi}$ instead of finding just the most likely solution, but even the computationally very efficient variant of hard matching works well in some cases ($N = 320$ gives 151 correct solutions). The factorized soft variant would not help here; it finds almost a delta distribution over the permutations and consequently also the resulting accuracy is comparable to the hard variant (not shown).

To illustrate the importance of good initialization, Figure 3 also shows the distribution of matching accuracies for each of the individual preliminary runs for the NF-Soft variant with $N = 320$ (the plot would be similar for other values of N). The accuracy varies considerably from one run to another and the final accuracy is better than any of the results obtained in the 50 initial runs, showing that a consensus of multiple runs provides an excellent initialization. While the initialization comes with a notable extra computational cost, the comparison with KS p-smooth is fair as both algorithms include an initialization

1. Note that the subsets of images used for $N < 320$ may not be exactly the same that were used by Jagarlamundi et al. (2010), so only the case of $N = 320$ is directly comparable.

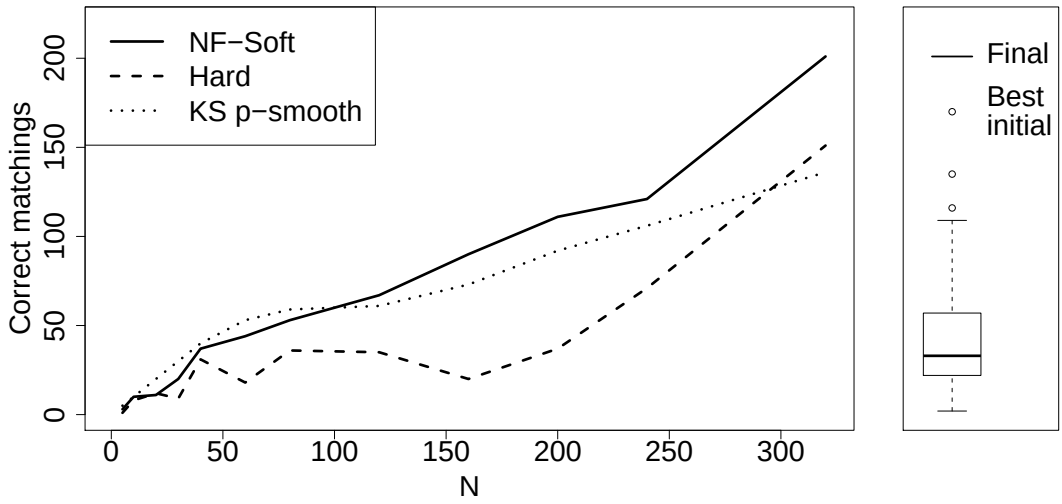

Figure 3: **Left:** The number of correct matches as a function of the number of samples N. The NF-Soft variant clearly outperforms the hard matching, and for all but the smallest data sets is also better than the state-of-art kernelized sorting method p-smooth. The results for p-smooth are measured with a ruler from Jagarlamundi et al. (2010). **Right:** For $N = 320$ the soft variant finds 201 correct matches. The boxplot shows the distribution of accuracies for the 50 initial runs with different initialization, demonstrating how the final consensus match is clearly better than any of these. In particular, the median accuracy of the initial runs is only 33, which means that we could not rely on any single initialization.

scheme that requires solving the match several times. To be on the safe side we used 50 initializations, compared to 200 runs by Jagarlamundi et al. (2010).

4.2. Metabolite Matching

Next we proceed to an example data on translational medicine, taken from Tripathi et al. (2011), where the task is to match metabolites of two populations. The problem mimics a challenge where we need to align metabolites of two different species (Sysi-Aho et al., 2011), but here the two populations are both human to provide the ground-truth alignment. The data consists of time series of concentrations of $N = 53$ metabolites, and we have measurements for several subjects. We compare our method with two methods presented by Tripathi et al. (2011), using a setup very similar to theirs. In particular, we average the matching accuracies over 100 runs where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are taken from random subjects (that is, the runs are truly independent since the input data is different in each run), and we restrict the matchings so that a metabolite can only pair with another one in the same functional class (which are assumed known).

We also provide another set of results without constraints, to demonstrate how well we can do without any prior information on the match. For both tasks we compare our

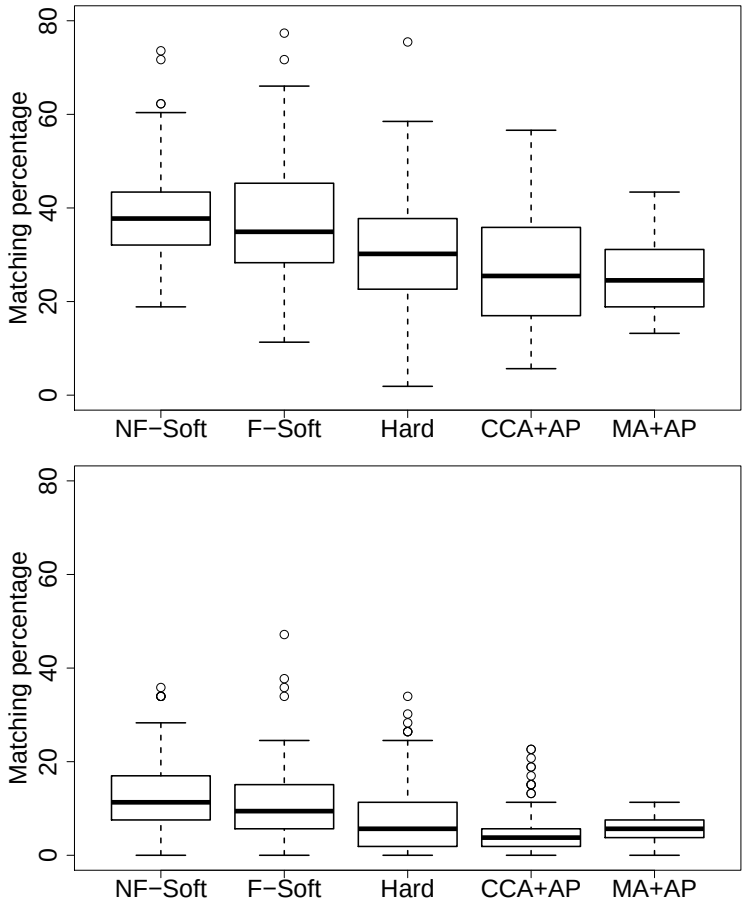

Figure 4: The accuracy of matching the metabolites in two human populations, averaged over 100 different matching tasks and summarized as boxplots using the default parameters of the R implementation. The **top** plot summarizes the results for setup where the matches are constrained to be within the known functional classes of the metabolites, whereas the **bottom** plot shows the results for purely data-driven solution with no additional constraints. For both setups the soft variants (both the non-factorized variant "NF-Soft" and the factorized one "F-Soft") of the proposed method are the best, followed by the hard variant. All three beat the baseline methods of CCA-based matching (CCA+AP) and manifold alignment (MA+AP) (paired t-test, $p < 0.05$).

method with the alternating optimization of CCA and AP (Tripathi et al., 2011) and manifold alignment (Wang & Mahadevan, 2009) followed by AP. The kernelized sorting is left out since Tripathi et al. (2011) already showed that both CCA+AP and MA+AP outperform the basic version, and we did not have an implementation of the p-smooth version by Jagarlamundi et al. (2010) at hand.

Figure 4 shows how we again outperform the earlier methods by a relatively wide margin[2], the non-factorized soft variant again having almost 50% more correct matches compared to the best earlier method (38.9% correct vs 26.9% correct for CCA+AP). Both of the soft variants provide comparable accuracy, outperforming the hard matching solution. Nevertheless, even the hard one outperforms the comparison methods. This is understandable since the hard version is very close to the CCA+AP model. Both learn the most likely matching with an AP solver, so the primary difference is in our method using the Bayesian CCA instead of classical CCA as the underlying model. Given the relatively small sample size of $N = 53$, it was to be expected that Bayesian CCA would outperform the classical one.

Note that in this experiment we did not use the advanced initialization strategy of learning the final model given a consensus of preliminary runs, to warrant fair comparison with the MA+AP model that could not utilize the same strategy as it does not require initialization. Instead, we merely initialized each run with the first PCA component. However, we did one final test to mimic the consensus matching setup of Tripathi et al. (2011), and found the consensus of the 100 runs with different input matrices to reach 85%, compared to their result of 70% with equal amount of data and some additional biological constraints not used in our solution.

5. Discussion

We introduced a variational Bayesian solution for the matching problem introduced by Jebara (2004) and popularized by Haghighi et al. (2008), Quadrianto et al. (2010) and Tripathi et al. (2011) for solving alignment tasks for example in natural language processing and computational biology. By learning together a Bayesian canonical correlation analysis model (Virtanen et al., 2011) and a permutation matrix re-ordering the samples in one of the sets, we obtained matching accuracies better than those of any earlier solution.

The good performance can be pinpointed to three key properties. First, using CCA as the underlying probabilistic model enables matching the samples in the subspace capturing only the dependencies between the sets. This observation has been noted also by the earlier solutions, which have explicitly maximized a dependency measure.

The second key element is that we learn a full distribution over the permutation matrices, instead of finding just the best one. Even if the overall goal was to find a single match, making the inference with soft assignments improves the accuracy considerably. In addition, it automatically implements properties similar to some earlier heuristic additions: Both Haghighi et al. (2008) and Jagarlamundi et al. (2010) attempt to identify the most reliable matches to trust them more, whereas we will automatically average over the unreliable matches by having wider posterior for them.

Finally, the third critical improvement is our improved initialization which dramatically helps converging to a good local optimum. The earlier works have also pointed out the importance of good initialization, and Jagarlamundi et al. (2010) in particular ended up devising an advanced initialization scheme that involves sequentially solving the problem with slightly different tuning parameter while starting with the best solution obtained thus far.

2. The exact values for the comparison methods are not directly comparable with Tripathi et al. (2011), since they used additional biological constraints for learning the match.

Our approach is complementary to that, by trying out several different initializations and then learning a consensus of those to be used as an initialization for the final solution. The proposed initialization is directly applicable also for other matching algorithms, including kernelized sorting.

During the review process Djuric et al. (2012) introduced a convex relaxation to kernelized sorting, obtaining accuracies comparable to ours in the image matching task. They got 206 correct matches for the 320 images, compared to our results of 201 ($K = 8$) and 216 ($K = 16$). Their method has computational advantage due to guaranteed convergence to a global optimum, but it lacks justified probabilistic treatment. They use soft assignments obtained by relaxation to doubly stochastic matrices, which means they do not necessarily solve the original problem, in contrast to our method that directly operates on true permutations. However, their solution would most likely be an excellent initialization for the proposed model, either as a replacement for the different random initializations or as one extra initialization to be included in the consensus.

On a broader perspective, we introduced techniques for dealing with permutations in variational approximation, to complement the earlier exact approaches for small sample sizes (Kondor et al., 2007; Plis et al., 2011). The ideas should be applicable also to other problems involving permutation matrices, such as object tracking (Kondor et al., 2007), assuming the mode (the optimal permutation) can be found efficiently. The algorithms for creating the alternative permutations could be improved, but already the current ones dramatically improve the accuracy compared to using only the best match. We also pointed out that breaking the independence assumption between the match and the latent variables is crucial for high-dimensional data.

Acknowledgements

We thank Academy of Finland (decision number #133818 and the Finnish Center of Excellence for Computational Inference (COIN)) for funding the research, and provide our grateful thanks for Prof. Matej Orešič for providing the data used in the metabolomics experiment.

References

Bach, F.R. and Jordan, M.I. A probabilistic interpretation of canonical correlation analysis. Technical Report 688, Department of Statistics, University of California, Berkeley, 2005.

Boyd-Graber, J. and Blei, D M. Multilingual topic models for unaligned text. In *Uncertainty in Artificial Intelligence*, 2009.

Djuric, N., Grbovic, M., and Vucetic, S. Convex kernelized sorting. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 893–899, 2012.

Haghighi, A., Liang, P., Berh-Kirkpatrick, T., and Klein, D. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-08: HLT*, pp. 771–779, 2008.

Jagarlamundi, J., Juarez, S., and Daumé III, H. Kernelized sorting for natural language processing. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)*, pp. 1020–1025, 2010.

Jebara, T. Kernelized sorting, permutation, and alignment for minimal volume PCA. In *Conference on Computational Learning Theory (COLT)*, volume 3120 of *LNAI*, pp. 609–623, 2004.

Klami, A. and Kaski, S. Local dependent components. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pp. 425–432, 2007.

Kondor, R., Howard, A., and Jebara, T. Multi-object tracking with representations of the symmetric group. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., and Ghahramani, Z. Kronecker graphs: an approach to modeling networks. *Journal of Machine Learning Research*, 11: 985–1042, 2010.

Plis, S. M., McCracken, S., Lane, T., and Calhoun, V. D. Directional statistics on permutations. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 600–608, 2011.

Quadrianto, N., Smola, A. J., Song, L., and Tuytelaars, T. Kernelized sorting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1809–1821, 2010.

Smola, A. J., Gretton, A., Song, L., and Schölkopf, B. A Hilbert space embedding for distributions. In *Algorithmic Learning Theory*, volume 4754 of LNCS, pp. 13–31, 2007.

Sysi-Aho, M. et al. Metabolic regulation in progression to autoimmune diabetes. *PLoS Computational Biology*, 7:e1002257, 2011.

Tripathi, A., Klami, A., Orešič, M., and Kaski, S. Matching samples of multiple views. *Data Mining and Knowledge Discovery*, 23:300–321, 2011.

Virtanen, S., Klami, A., and Kaski, S. Bayesian CCA via group sparsity. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 457–464, 2011.

Wang, C. Variational Bayesian approach to canonical correlation analysis. *IEEE Transactions on Neural Networks*, 18:905–910, 2007.

Wang, C. and Mahadevan, S. Manifold alignment without correspondence. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1273–1278, 2009.