Mach Learn (2013) 92:225–250
DOI 10.1007/s10994-013-5357-4

Bayesian object matching

Arto Klami

Received: 14 January 2013 / Accepted: 11 April 2013 / Published online: 30 April 2013

Abstract Matching of object refers to the problem of inferring unknown co-occurrence or alignment between observations or samples 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. Given a distance measure, the problem would correspond to a linear assignment problem, the problem of finding a permutation that re-orders samples in one set to minimize the total distance. When no such measure is available, we need to consider more complex solutions. Typical approaches maximize statistical dependency between the two sets, whereas in this work we present a Bayesian solution that builds a joint model for the two sources. We learn a Bayesian canonical correlation analysis model that includes a permutation parameter for re-ordering the samples in one of the sets. We provide both variational and sampling-based inference for approximative Bayesian analysis, and demonstrate on three data sets that the resulting methods outperform the earlier solutions.

Keywords Canonical correlation analysis · Matching · Permutation · Bayesian analysis

1 Introduction

The task in object matching is to learn correspondence of samples in two data sets. A classical example considers a set of agents and another set of jobs, and the task is to assign each job for exactly one agent. For each agent-job pair we have a specific cost, corresponding, for example, to how well they perform the job or how much it costs, and the goal is to find the assignment that minimizes (or maximizes) the total assignment or matching cost. The problem is known as the maximum weight matching in a bipartite graph, or as linear assignment problem. Efficient polynomial time algorithms exist for finding the match, such as the classical Hungarian algorithm (Kuhn 1955).

Editors: Zhi-Hua Zhou, Wee Sun Lee, Steven Hoi, Wray Buntine, and Hiroshi Motoda.

A. Klami (✉)
Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki, Helsinki, Finland
e-mail: arto.klami@hiit.fi

The classical setup takes the costs for the agent-job assignments as input. This is equivalent to assuming that we are given a distance measure between the two sets, and the goal is to minimize the total distance. In this work we study more complicated setups where no such distance is known. Instead, we are merely given two vector data sets representing the items in the two sets, with no known relationship between the feature representations. In the classical example of agent-job matching, we might have for each agent a set of features describing their physical abilities and test scores, whereas for the jobs we could have the text of the job advert. The task is still to assign each job for one of the agents, but obviously the standard algorithms for solving the assignment problem do not apply. To highlight the notion of possibly wildly different feature representations, Yamada and Sugiyama (2011) used the phrase cross-domain object matching (CDOM) to denote the problem.

Since no distance between the sets is given, we must replace the task of minimizing the total distance by something else. In recent years, a number of solutions have been proposed based on maximization of statistical dependency, measured for example by the mutual information, between the two sets. This corresponds to making an assumption that the correct assignment is the one that reveals the highest degree of statistical dependency between the sets. One intuitive justification for the idea is the observation that randomly permuting samples of the correct match will decrease the dependency, eventually making the two sets independent if all pairs are replaced with random ones. To our knowledge, the idea of finding the match that maximizes the statistical dependency was independently first suggested by Haghighi et al. (2008), Tripathi et al. (2009), and Quadrianto et al. (2009). The first two measure the dependency with linear canonical correlations, whereas the last one uses the kernel-based Hilbert-Schmidt Independence Criterion (HSIC; Smola et al. 2007), but conceptually the methods are closely related. Later for example Tripathi et al. (2010), Tripathi et al. (2011), Jagarlamudi et al. (2010), Quadrianto et al. (2010), Yamada and Sugiyama (2011) and Djuric et al. (2012) have presented improved models and extensions based on the same idea.

An alternative criterion for learning the match comes from joint modeling. Given the two sets, we can write a joint generative model for both, including a permutation over the samples of one of the sets as part of the model. Then a reasonable assumption is that the correct match is obtained with the permutation that results in the best joint model. This idea was proposed before the dependency-maximization solutions by Jebara (2004) who maximized the likelihood of a joint Gaussian model. Another example of a method optimizing the joint likelihood is the matching canonical correlation analysis (MCCA) by Haghighi et al. (2008), which was already mentioned in the previous paragraph; it can be seen as a method that both maximizes the canonical correlation between the two sets but also maximizes the likelihood of a specific probabilistic model, namely the probabilistic interpretation of canonical correlation analysis (CCA; Bach and Jordan 2005).

In this work we present another solution that fits both motivations, extending our preliminary publication (Klami 2012). We solve the matching problem by building an optimal joint model, but instead of maximizing the likelihood we do full Bayesian inference, searching for a posterior distribution over the permutations to characterize the set of possible matches. The connection to dependency-maximizing solutions is obtained by choosing Bayesian canonical correlation analysis (BCCA; Klami et al. 2013) as the underlying probabilistic model. Since the model implements CCA, our solution will also maximize the canonical correlation between the sets. Furthermore, it corresponds to the Bayesian solution of MCCA.

The match is learned by introducing a permutation parameter $\boldsymbol{\pi}$, which is a $N \times N$ binary matrix with unit row and column sums, into the BCCA model. The main challenge in Bayesian analysis of the model is then in learning the posterior over the permutations. While

the set of permutation matrices is discrete, there are N factorial different permutations. Exact inference over such a huge space is not feasible, and hence the main contributions of this work are several alternative approximative strategies that enable simultaneous posterior inference over the permutations and the rest of the BCCA parameters. In particular, we will present both a Gibbs sampler with approximative and exact conditional distributions for the permutations given the rest of the parameters, as well as variational approximation with varying degree of accuracy for a term approximating the posterior over the permutations. The former is a novel contribution of this work, whereas the latter was preliminary presented already by Klami (2012).

To empirically evaluate the various alternative approximations, we compare the proposed method with the best existing algorithms using three benchmark data sets: matching left and right sides of images based on their content, matching metabolic profiles of different individuals, and cross-lingual document alignment. For all experiments we compare the proposed methods with the leading kernelized sorting variant of convex kernelized sorting (CKS) by Djuric et al. (2012) and the maximum likelihood solution of our model, which corresponds to Haghighi et al. (2008) and Tripathi et al. (2011). In all three tasks the proposed methods outperform the earlier methods.

We will start by formally introducing the matching problem and our Bayesian formulation for it. We then summarize the Bayesian canonical correlation analysis (BCCA) model as presented by Klami et al. (2013), which is used as the underlying latent variable model in our matching solutions. We go through both the sampling and variational inference for that model, and then proceed to the main contributions of this work: The matching Bayesian CCA. Again we cover both the sampling and variational inference, before explaining related work and the empirical comparisons.

2 Object 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 $N \times N$ permutation matrices, binary matrices with unit row and column sums.

In this work the samples will be represented as real-valued vectors, and hence the sets are represented as matrices $\mathbf{X} \in \mathbb{R}^{D_x \times N}$ and $\mathbf{Y} \in \mathbb{R}^{D_y \times N}$. Individual samples will be denoted by column vectors $\mathbf{x}_i$ and $\mathbf{y}_j$. In general, there does not need to be any correspondence between the feature spaces of the two sets; their dimensionalities can differ, as well as the actual features.

In case we have a distance measure between the two sets, providing distances d_{ij} between the samples $\mathbf{x}_i$ and $\mathbf{y}_j$, the problem is straightforward. It is an instance of bipartite graph matching problems, and can be solved as a linear assignment problem (Kuhn 1955) by minimizing the total distance $c(\mathcal{X}, \mathcal{Y}|\boldsymbol{\pi}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{\pi}_{ji} d_{ij}$. In this work we are interested in scenarios where no such distance is known. Then we need to use alternative costs that, when maximized, result in a good match.

2.2 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

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

That is, the i.i.d. samples in the two data sets are conditionally independent given a latent variable $\mathbf{z}_i$.

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 $\mathbf{y}_j$ pairs with the latent variable $\mathbf{z}_i$ (which is associated with the sample $\mathbf{x}_i$), and using $\boldsymbol{\pi}_{.i}$ to denote the ith column we can write the Bayesian matching model as

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

The task in Bayesian matching is then to find the posterior distribution of the permutation, $p(\boldsymbol{\pi}|\mathbf{X}, \mathbf{Y})$, which cannot be done analytically. In this work we will introduce two alternative strategies for approximating it for one particular model. One approach is based on variational approximation of the posterior and the other uses Gibbs sampling to draw samples from the posterior. While any joint model could in principle be used, the inference details will depend on the choice of the model. Next, we will summarize our choice for the underlying model before explaining how it needs to be modified to solve the matching problem.

3 Bayesian CCA

As the actual model we use the Bayesian CCA model as presented by Klami et al. (2013), 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, since CCA uses separate components independent of the permutation for modeling that.

The Bayesian CCA is a fairly simple linear model for two multivariate data sets. Each sample is represented by a latent variable $\mathbf{z}_i$ which is linearly transformed to both observation spaces, complemented with additive Gaussian noise. The Bayesian CCA for K components is defined as

$$\begin{aligned} \mathbf{z}_i &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \\ [\mathbf{x}_i; \mathbf{y}_i] &\sim \mathcal{N}(\mathbf{W}\mathbf{z}_i, \boldsymbol{\Sigma}), \end{aligned} \tag{2}$$

where $[\mathbf{x}_i; \mathbf{y}_i]$ denotes the feature-wise concatenation of the samples with $D = D_x + D_y$ dimensions and $\mathbf{z}_i \in \mathbb{R}^{K\times 1}$. The basic idea of BCCA is that the latent components model only the correlations. To achieve this, we either need to use block-diagonal covariance $\boldsymbol{\Sigma}$

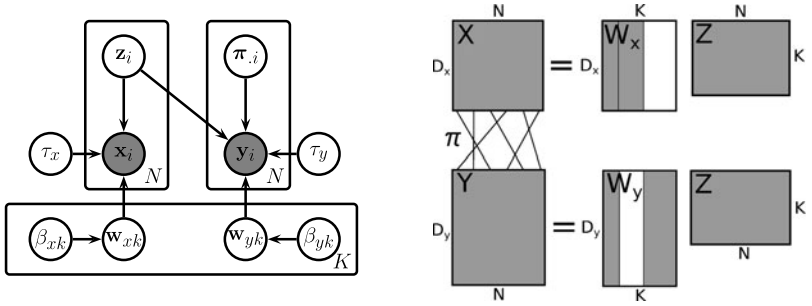

Fig. 1 *Left*: Plate diagram of the model. The latent variables $\mathbf{z}_i$ are directly associated with samples $\mathbf{x}_i$, whereas the samples $\mathbf{y}_i$ use the permutation $\boldsymbol{\pi}$ to choose which latent variable to use. The ARD prior terms β_{xk} and β_{yk} induce sparsity in the linear mappings $\mathbf{W}_x$ and $\mathbf{W}_y$ so that some of the K components are being used for modeling variation independent of the other data set. The plate corresponds to the ARD prior used with variational approximation; the spike-and-slab prior used for the Gibbs sampler has the same general structure and differs only in the priors given for $\mathbf{W}$. *Right*: Illustration of the model as coupled matrix factorization. The two factorizations are tied through the shared set of latent variables $\mathbf{Z}$ and through the permutation $\boldsymbol{\pi}$ re-ordering the observations in $\mathbf{Y}$ (which is equivalent to re-ordering latent variables for that set). The projection matrices $\mathbf{W}_x$ and $\mathbf{W}_y$ are made sparse (the un-shaded regions correspond to zero values) so that some of the components are used for modeling variation shared between the two sets, whereas some model the variation independent of the other set

that allows free correlations between the features within each set (Bach and Jordan 2005; Klami and Kaski 2007), or we can use diagonal covariance

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

but need to model the view-specific correlations with additional components (Virtanen et al. 2011; Klami et al. 2013); this equals assuming the view-specific variation can be modeled with low-rank covariance.

Here we adopt the latter choice, which results in considerably faster and more accurate algorithm for high-dimensional data. In particular, we impose group-wise sparsity prior on $\mathbf{W} \in \mathbb{R}^{D \times K}$ so that some of the K components are used for modeling dependencies between the two sets, whereas some are used for describing variation independent of the other set. By re-writing $\mathbf{W} = [\mathbf{W}_x; \mathbf{W}_y]$ so that $\mathbf{W}_x$ covers the dimensions spanned by $\mathbf{X}$ (and similarly for $\mathbf{W}_y$), we want solutions where some of the columns of $\mathbf{W}$ become sparse in a very specific sense: Either $\mathbf{W}_x$ or $\mathbf{W}_y$, or both, is completely zero for that component.

Such a solution splits the components into three groups. For some components all elements are free; these are the components that model the correlations between the two sets. For some components the elements in $\mathbf{W}_x$ are free but the elements in $\mathbf{W}_y$ are zero; these model variation specific to the data set $\mathbf{X}$. Similarly, the components for which $\mathbf{W}_y$ is free but $\mathbf{W}_x$ is zero model the variation specific to $\mathbf{Y}$. Finally, components for which both parts become zero can be dropped out, resulting in automatic complexity selection.

The model is illustrated in Fig. 1, which shows the plate diagram of the generative mode, as well as graphical illustration of how the group-wise sparsity makes some of the components to model low-rank covariance specific to each data set. The figure shows the model as it is used for matching, explained in Sect. 4, but dropping the permutation makes the illustration applicable to regular BCCA as well. Next we will briefly recap two alternative in-

ference algorithms for BCCA following Klami et al. (2013), using two different approaches for achieving the group-wise sparsity structure, before explaining the matching variant.

3.1 Gibbs sampling with spike-and-slab prior

The spike-and-slab is a mixture prior that gives some probability for the value to be exactly zero (the spike) and some probability for it to be drawn from a proper distribution (the slab). We use a group-wise extension of spike-and-slab by drawing for each component two binary latent variables $\mathbf{h}_{xk}$ and $\mathbf{h}_{yk}$ that tell whether to draw the values of $\mathbf{W}_{xk}$ and $\mathbf{W}_{yk}$ from the slab (a Gaussian distribution) or from the spike (delta distribution at zero). Here $\mathbf{W}_{xk}$ denotes the D_x-dimensional vector corresponding to the kth component, the kth column of $\mathbf{W}_x$. The $\mathbf{h}$ are drawn from Bernoulli distributions, and for $\mathbf{h}_{xk} = 1$ we then draw the elements of $\mathbf{W}_{xk}$ independently from a Gaussian distribution whose precision β_{xk} is sampled from a Gamma prior. For $\mathbf{h}_{yk} = 0$ we simply set all elements of $\mathbf{W}_{xk} = 0$ to zero, and the formulas for $\mathbf{Y}$ are equivalent except for the subscript.

Klami et al. (2013) derived a Gibbs sampler for BCCA using the above prior, by extending the element-wise sparse factor analysis model by Knowles and Ghahramani (2011). The algorithm is conceptually very straightforward, drawing each parameter from its analytic posterior given all of the other parameters. Since the model uses conjugate priors for all of the parameters, these can be derived easily. The latent variables $\mathbf{h}$, however, are drawn by integrating out $\mathbf{W}$; the resulting posterior is still analytically tractable. The full conditional densities are summarized in the Appendix.

3.2 Variational approximation with ARD prior

An alternative for sampling is to approximate the posterior distribution with a simpler, tractable, distribution. The approximation is learned by minimizing the Kullback-Leibler divergence between the approximation and the true posterior. We adopt the variational approximation provided by Klami et al. (2013), which uses group-wise extension of the automatic relevance determination (ARD) prior for achieving the right kind of sparsity.

The prior is defined as

$$\beta_{xk} \sim \mathcal{G}(\alpha_0, \beta_0), \qquad \beta_{yk} \sim \mathcal{G}(\alpha_0, \beta_0),$$

$$p(\mathbf{W}) = \prod_{k=1}^{K} \left(\mathcal{N}\left(\mathbf{W}_{xk}|\mathbf{0}, \beta_{xk}^{-1}\mathbf{I}\right)\mathcal{N}\left(\mathbf{W}_{yk}|\mathbf{0}, \beta_{yk}^{-1}\mathbf{I}\right)\right).$$

where $\mathbf{W}_{xk}$ again denotes the kth column of $\mathbf{W}_x$, and $\mathcal{G}(\alpha_0, \beta_0)$ is a flat Gamma prior. It achieves the group-wise sparsity by driving the precision β_{xk} towards infinity for the components that are not needed for modeling the $\mathbf{X}$ data set (and similarly for $\mathbf{Y}$). Contrary to the spike-and-slab prior, this does not result in exact zeroes in $\mathbf{W}$. Instead, the unnecessary values will just become very small; this is still sufficient for the model to provide the CCA solution, and it is easier to do variational approximation over a continuous prior.

An efficient variational approximation is provided by the factorization

$$Q = q(\tau_x)q(\tau_y)\prod_{k=1}^{K} q(\beta_{xk})q(\beta_{yk})\prod_{i=1}^{N} q(\mathbf{z}_i)\prod_{d=1}^{D} q(\mathbf{W}_{d.}), \tag{3}$$

and the parameters of each term are learned by updating alternatively each of the term. The full updates are given by Klami et al. (2013), and are hence not replicated here.

4 Matching Bayesian CCA

The matching Bayesian CCA model extends the above formulations by replacing the likelihood part in (2) with

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

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 resulting model is illustrated in Fig. 1.

Including the permutation in the model requires also corresponding changes in the inference procedures. These changes are conceptually very easy. For the sampler we merely include the new parameter in the model, derive a distribution for sampling it given the rest of the parameters, and condition all other sampling formulas on the permutation. For the variation approximation we complement (3) with an extra term $q(\boldsymbol{\pi})$, which is a distribution over permutation matrices, and an update rule for that term. Also, we need to change the updates for other terms to integrate over the new term.

In both approaches most of these changes are easy to do. However, the part where we either sample the permutation or update its approximation is non-trivial. We will next present several alternative techniques for achieving these steps, starting with the Gibbs sampler.

5 Matching BCCA with Gibbs sampler

Given a specific permutation, the changes needed for the sampling equations of $\mathbf{W}$, $\mathbf{h}$, β and τ are trivial; we merely need to re-order the samples in $\mathbf{Y}$ by multiplying it with the current permutation $\boldsymbol{\pi}$ before drawing the new samples.

The two remaining parameters, the permutation $\boldsymbol{\pi}$ and the latent variables $\mathbf{z}_i$ depend cyclicly on each other, and hence we sample them jointly, based on a slight reparameterization of the model. The latent variables $\mathbf{z}_i$ can be equivalently written as

$$\mathbf{z}_i = \boldsymbol{\mu}_i^x + \boldsymbol{\mu}_j^y + \boldsymbol{\xi}_i$$

where $\boldsymbol{\mu}_i^x = \boldsymbol{\Sigma}_z \tau_x \mathbf{W}_x^T \mathbf{x}_i$, $\boldsymbol{\mu}_j^y = \boldsymbol{\Sigma}_z \tau_y \mathbf{W}_y^T \mathbf{y}_j$, and $\boldsymbol{\xi}_i$ is zero-mean Gaussian noise with covariance $\boldsymbol{\Sigma}_z = (\tau_x \mathbf{W}_x^T \mathbf{W}_x + \tau_y \mathbf{W}_y^T \mathbf{W}_y + \mathbf{I})^{-1}$. In other words, the posterior distribution depends on two deterministic terms, one that depends on the sample $\mathbf{x}_i$ and the other that depends on the corresponding sample $\mathbf{y}_j$ in the other set, as well as an independent noise term. Importantly, the random term does not depend on the permutation at all; changing the permutation only influences the second deterministic term.

Using $\boldsymbol{\Xi}$ to denote the collection of all $\boldsymbol{\xi}_i$ variables, we can write the conditional distribution $p(\boldsymbol{\pi}, \mathbf{Z}|\text{rest})$ as $p(\boldsymbol{\pi}, \boldsymbol{\Xi}|\text{rest}) = p(\boldsymbol{\pi}|\boldsymbol{\Xi}, \text{rest}) p(\boldsymbol{\Xi}|\text{rest})$, decoupling the two variables. Here "rest" indicates all of the other model parameters, which are not explicitly written for notational simplicity. In principle it would be easy to draw samples from this conditional; the latter term is Gaussian and the former is a discrete distribution for which we can evaluate the log-probabilities as

$$\log p(\boldsymbol{\pi}|\boldsymbol{\Xi}) = -\sum_{i=1}^{N} \tau_y (\mathbf{y}_j - \mathbf{W}_y \mathbf{z}_i)^T (\mathbf{y}_j - \mathbf{W}_y \mathbf{z}_i) + \text{const},$$

where j is chosen such that $\boldsymbol{\pi}_{ji} = 1$. However, for directly drawing posterior samples from this discrete distribution we would need to normalize the probabilities with a term that requires computing the probability for all possible permutations. This is infeasible for all but the smallest N, since there are $N!$ different permutations.

To handle the difficult conditional we propose an approximative Metropolis-Hastings step with Gibbs-like proposal distribution $q(\boldsymbol{\pi}^*, \boldsymbol{\Xi}^*) = q(\boldsymbol{\pi}^*|\boldsymbol{\Xi}^*)q(\boldsymbol{\Xi}^*)$ that does not depend on the current values of $\boldsymbol{\pi}$ and $\boldsymbol{\xi}$. For the latter term we use the actual conditional distribution $q(\boldsymbol{\Xi}^*) = p(\boldsymbol{\Xi}^*|\text{rest})$ that is a Gaussian, whereas for $q(\boldsymbol{\pi}^*|\boldsymbol{\Xi}^*)$ we use a simple proposal distribution that is the delta distribution centered around the most likely permutation given $\boldsymbol{\Xi}^*$ and the other parameters. We can easily find that permutation by solving a LAP with the cost matrix

$$\mathbf{A}_{ij} = \tau_y \big(\mathbf{y}_j - \mathbf{W}_y \mathbf{z}_i^*\big)^T \big(\mathbf{y}_j - \mathbf{W}_y \mathbf{z}_i^*\big). \tag{4}$$

The Metropolis-Hastings ratio for acceptance of the proposals is given by

$$\frac{p(\boldsymbol{\pi}^*, \boldsymbol{\Xi}^*)q(\boldsymbol{\pi}, \boldsymbol{\Xi})}{p(\boldsymbol{\pi}, \boldsymbol{\Xi})q(\boldsymbol{\pi}^*, \boldsymbol{\Xi}^*)} = \frac{p(\boldsymbol{\pi}^*|\boldsymbol{\Xi}^*)p(\boldsymbol{\Xi}^*)q(\boldsymbol{\pi}|\boldsymbol{\Xi})q(\boldsymbol{\Xi})}{p(\boldsymbol{\pi}|\boldsymbol{\Xi})p(\boldsymbol{\Xi})q(\boldsymbol{\pi}^*|\boldsymbol{\Xi}^*)q(\boldsymbol{\Xi}^*)} = \frac{p(\boldsymbol{\pi}^*|\boldsymbol{\Xi}^*)}{p(\boldsymbol{\pi}|\boldsymbol{\Xi})}$$

where $q(\boldsymbol{\pi}|\boldsymbol{\Xi}) = q(\boldsymbol{\pi}^*|\boldsymbol{\Xi}^*) = 1$ and $q(\boldsymbol{\Xi}) = p(\boldsymbol{\Xi})$ cancel out. Consequently, the proposals should be accepted with probability $\min(1, p(\boldsymbol{\pi}^*|\boldsymbol{\Xi}^*)/p(\boldsymbol{\pi}|\boldsymbol{\Xi}))$. Computing the ratio is infeasible due to the normalization constants that are sums over all permutations.

We propose approximating the ratio with a constant $p(\boldsymbol{\pi}^*|\boldsymbol{\Xi}^*)/p(\boldsymbol{\pi}|\boldsymbol{\Xi}) = 1$, which means accepting all proposals. The sampler will hence not draw samples from the true posterior, but in practice the approximation is fairly accurate due to two properties. First, the relative probability of the most likely permutation is effectively independent of the likelihood of that permutation. This means that the sampler does not impose bias in favor or against permutations that fit the data well. Instead, it merely has a small bias towards $\boldsymbol{\Xi}$ that would give roughly equal probability for several permutations, the kind of latent variable allocations where some variables lay near the Voronoi border of two data points. Second, for high-dimensional data $p(\boldsymbol{\pi}|\boldsymbol{\Xi})$ approaches one (and hence also the ratio approaches one), since the best permutation becomes increasingly more likely compared to all others. We provide empirical evidence for these arguments in Sect. 9.1. Using small enough N that makes explicit normalization of $p(\boldsymbol{\pi}|\boldsymbol{\Xi})$ feasible and hence permits exact Gibbs steps, we show that the resulting marginal posterior over the permutations is a good approximation for the true marginal posterior. An intuitive explanation is that even though we use a crude approximation for $q(\boldsymbol{\pi}|\boldsymbol{\Xi})$, the marginal posterior depends more on $\boldsymbol{\Xi}$ and hence the joint samples will still cover roughly the same part of the permutation space. This also makes the chain ergodic.

In case one is not willing to make the above approximation, an exact sampler can be derived by updating only a subset of latent variables at a time. For a subset of sufficiently small size (in practice at most 6 or 7) we can go though all possible permutations and compute the relative probabilities of each of them. We then draw that subset of the permutation matrix by conditioning on the rest of the parameters and the remaining part of the permutation. In practice we suggest drawing a random subset of samples, updating the posterior for those, and repeating this procedure several times to obtain the next posterior sample.

An alternative exact sampler could be derived based on the pseudo-marginal likelihood method by Andrieu and Robers (2009). They prove that the even if the likelihoods in Metropolis-Hastings acceptance ratio are replaced with estimates the chain will still produce samples from the correct posterior, assuming the estimation error is independent of

the sampling chain. In our case, we could estimate the likelihood $p(\boldsymbol{\pi}|\boldsymbol{\Xi})$ by estimating the normalization constant e.g. with random walks over the permutations. However, we do not think that the added computational overhead needed to estimate the normalization constant would be worth it, and hence we experiment only with the approximative variant. From this perspective, it corresponds to using a constant estimate $p(\boldsymbol{\pi}|\boldsymbol{\Xi}) \approx c$, which already seems to have almost independent approximation error.

Posterior summaries The samplers described above produce a collection of samples from the posterior distribution $p(\mathbf{Z}, \boldsymbol{\pi}, \tau, \mathbf{h}, \mathbf{W}, \beta|\mathbf{X}, \mathbf{Y})$. In matching problems the primary interest is in the marginal posterior $p(\boldsymbol{\pi}|\mathbf{X}, \mathbf{Y})$, which is obtained by counting how many times each of the possible permutations were drawn during the process.

Various posterior summaries can be useful for interpretation. The most obvious one is the mean over the permutations $\tilde{\boldsymbol{\pi}} = \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{\pi}^{(s)}$, where $\boldsymbol{\pi}^{(s)}$ is the sth posterior sample collected after a burn-in period and S is the total number of samples. This mean is not a permutation matrix itself, but instead a doubly-stochastic matrix that can be interpreted as a soft permutation; its entries tell the probability that two samples are paired with each other.

In many applications we are also interested in obtaining a single permutation that summarizes the whole posterior. We create the summary by finding the permutation $\boldsymbol{\pi}$ that maximizes $\sum \mathrm{diag}(\tilde{\boldsymbol{\pi}}\boldsymbol{\pi})$, the total probability of each sample pairing with its chosen pair. The cost is again that of an assignment problem, and hence we obtain the solution by applying LAP to the weight matrix $\tilde{\boldsymbol{\pi}}$. This solution is the same that Tripathi et al. (2011) used for finding a consensus of multiple isolated matching tasks.

6 Matching CCA with variational approximation

The variational approximation for matching CCA was presented in our preliminary work (Klami 2012). Again the basic idea is straightforward; we merely complement the posterior approximation with $q(\boldsymbol{\pi})$, which is a distribution over a set of the permutations.

Even approximating the distribution that is over the $N!$-dimensional space initially sounds infeasible, but in practice the problem is simplified by the same observations that were used to motivate the Gibbs sampler above: (i) We can efficiently find the permutation $\hat{\boldsymbol{\pi}}$ that maximizes the variational lower bound and (ii) only a tiny faction of other permutations have clearly non-zero probability. 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 $\mathbf{Y}$ by $\langle\boldsymbol{\pi}\rangle$ after updating the match. Hence, the only challenging part is again in updating the permutations.

6.1 The most likely permutation

For the Gibbs sampler we could find the most likely permutation by merely looking at the distances between $\mathbf{y}_j$ and $\mathbf{W}_y\mathbf{z}_i$. For variational approximation, however, we need to integrate over the approximating distribution. The integrals can be computed analytically since the approximation factorizes over the different terms, but the resulting formulas are a bit more complex than for the sampler.

The log-cost of a permutation is given by

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

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 $\mathbf{A}$ with entries

$$\begin{aligned}\mathbf{A}_{ij} &= \frac{1}{2}\left\langle \tau_y (\mathbf{y}_j - \mathbf{W}_y\mathbf{z}_i)^T (\mathbf{y}_j - \mathbf{W}_y\mathbf{z}_i)\right\rangle \\ &= \frac{1}{2}\langle\tau_y\rangle\left[\mathbf{y}_j^T\mathbf{y}_j - 2\mathbf{y}_j^T\langle\mathbf{W}_y\rangle\langle\mathbf{z}_i\rangle + \mathrm{Tr}\left(\left\langle\mathbf{W}_y^T\mathbf{W}_y\right\rangle\left\langle\mathbf{z}_i\mathbf{z}_i^T\right\rangle\right)\right], \end{aligned} \tag{5}$$

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 $\sum \mathrm{diag}(\mathbf{A}\hat{\boldsymbol{\pi}})$ is minimized.

Similar to the re-parameterization in the Gibbs sampler, the values $\langle\mathbf{z}_i\rangle$ are based on a representation that de-couples the permutation and the stochastic part. This time the relevant expression stems from the update rule for $q(\mathbf{z}_i) = \mathcal{N}(\boldsymbol{\mu}_{ij}, \boldsymbol{\Sigma}_z)$ that can be written as

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

Again the permutation only influences the second deterministic term in the mean $\boldsymbol{\mu}_{ij}$. This re-parameterization allows writing the distances in the compact form (5) and also makes transparent an additional factorizing assumption: we assume that the uncertain part $\boldsymbol{\xi}_i = \mathbf{z}_i - \boldsymbol{\mu}_{ij}$ (with covariance $\boldsymbol{\Sigma}_z$) is independent of j. This is a necessary assumption for efficient computation, but means that the approximation is more restricted.

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 $\mathbf{z}_i$ for the components that do not describe any variation in $\mathbf{X}$ (that is, β_{xk} is very large) – the ones not active in $\mathbf{Y}$ anyway have no effect on $\mathbf{A}$. Denoting by $\mathbf{V}_y$ the columns of $\mathbf{W}_y$ corresponding to the components marginalized out and by $\mathbf{U}_y$ the remaining columns, the entries in (5) are replaced by $\frac{1}{2}\langle(\mathbf{y}_j - \mathbf{U}_y\mathbf{z}_i)\boldsymbol{\Psi}(\mathbf{y}_j - \mathbf{U}_y\mathbf{z}_i)\rangle$, where $\boldsymbol{\Psi} = (\mathbf{V}_y\mathbf{V}_y^T + \tau_y^{-1}\mathbf{I})^{-1}$. For efficient inversion we use the Woodbury identity $\boldsymbol{\Psi} = \tau_y\mathbf{I} - \tau_y^2\mathbf{V}_y(\mathbf{I} + \tau_y\mathbf{V}_y^T\mathbf{V}_y)^{-1}\mathbf{V}_y^T$. For large D_y we can further approximate $\boldsymbol{\Psi} = \tau_y\mathbf{I}$ for faster computation, since the matrix is anyway close to diagonal in many cases.

6.2 Full posterior over the 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.

Local perturbations By making the assumption that $\boldsymbol{\xi}_i$ are independent of the permutation $\boldsymbol{\pi}$, we can directly use costs of the form $w_m \propto e^{-\sum \mathrm{diag}(\mathbf{A}\boldsymbol{\pi}^{(m)})}$ to obtain the relative 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 local unimodal approximation centered around the mode, to complement the other unimodal terms in (3).

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 $\mathbf{A}$ 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 high-dimensional data. Hence, for high-dimensional data it might be feasible to ignore the alternative permutations altogether and simply use the most likely one. 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 a local 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. 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.

Numerical integration Like explained above, the straightforward variational approximation makes one fairly strong independence assumption: It assumes that $\boldsymbol{\xi}_i$, the stochastic parts of the latent variables $\mathbf{z}_i$, are independent of the match. This assumption allows efficient computation of $\mathbf{A}_{ij}$, but it also implies that the relative probabilities of the different permutations will be incorrect. In particular, the uncertainty in the distances is not fully taken into account, but instead the model favors strongly the closest samples even if the variance of the distance was large.

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

Explicitly modeling such dependencies requires simultaneously integrating over the N latent variables $\mathbf{z}_i$ to compute the relative costs of different permutations. Solving such an integral analytically seems hopeless, and hence we resort to Monte Carlo integration which closely resembles the way the Gibbs sampler works. Instead of assuming independent set of $\boldsymbol{\xi}_i$ values for each sample $\mathbf{y}_j$ and integrating them out (which would correspond to (5)), we draw a joint random sample $\boldsymbol{\Xi}^{(m)} = \{\boldsymbol{\xi}_i^{(m)}\}_{i=1}^{N}$ that is independent of j. We then recompute $\mathbf{A}_{ij}$ for all pairs, not integrating over $\mathbf{z}_i$ but instead replacing them with the sampled value (note, however, that we still integrate over $q(\tau_y)$ and $q(\mathbf{W})$). We then find the best

permutation for this particular sample, and by repeating the 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.

6.3 Initialization

As demonstrated by most earlier works, the matching problem is very sensitive to the initialization. Our variational 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 get a set of different but still good initializations, 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. Alternatively, one could add some random noise to the first PCA component before sorting the data points. Finally, we make the initial permutation smoother by convolving it with a Gaussian kernel (the exact width does not seem to matter).

7 Related work

Most earlier solutions to the matching problem maximize statistical dependency between the two sets. The idea is that statistical dependency should not arise by coincidence, but instead a high degree of dependency should be indicative of having found the correct match. A random permutation will make any two sets independent, and hence maximizing the dependency will at least allow escaping that extreme. The practical methods can be divided into two categories based on the dependency measure. The first category optimizes canonical correlation between the sets, whereas the other category maximizes a kernel-based dependency called Hilbert-Schmidt Independence Criterion (HSIC; Smola et al. 2007) or some other kernel-based measure. The former require access to real-valued feature vectors for the two sets, whereas for the latter it is sufficient to provide kernels representing pairwise-distances within each set.

The matching canonical correlation analysis (MCCA) method was introduced by Haghighi et al. (2008) for constructing bilingual dictionaries from monolingual corpora, by matching the individual words in two languages. The basic idea of the algorithm is that

it finds a linear subspace that maximizes the correlation between the sets (CCA; canonical correlation analysis). Explicit representation for the subspace allows computing distances between the samples in the two sets, and hence a LAP can be used for finding the match. Since the subspace itself depends on the match, the algorithm alternates between these two steps until convergence. The original algorithm is defined for semi-supervised matching that requires an initial seeding with some known pairs, but Tripathi et al. (2009) presented independently almost the same algorithm for fully unsupervised matching of probes of gene expression platforms, and Tripathi et al. (2010) extended it to use kernel CCA while also presenting the semi-supervised matching problem where some initial seed pairs are given. Later, Tripathi et al. (2011) extended the CCA-based formulation to setups where the task is to find a consensus of multiple matching problem solutions, and Sysi-Aho et al. (2011) applied it to finding correspondence between metabolic profiles of different species. The basic idea is to merge several matching problem solutions by learning one more LAP to find a permutation that best agrees with the initial solutions.

The other category of dependency-maximizing matching solutions builds on the kernelized sorting (KS) idea initially presented by Jebara (2004). Instead of finding an explicit representation that allows computing distances (and hence solving the matching as a LAP), the idea in kernelized sorting is to directly optimize a dependency measure that only depends on kernels computed for the two sets. Quadrianto et al. (2010) introduced the standard KS algorithm that maximizes the Hilbert-Schmidt Independence Criterion. The HSIC is defined as the trace-norm of KL, where K and L are properly centered kernel matrices for the two sets, and KS solves the matching problem by introducing a permutation matrix in that cost. The resulting cost corresponds to a quadratic assignment problem (QAP), which is NP complete (Burkard 1984). Quadrianto et al. (2010) solve the QAP by iteratively applying a LAP solver. Later Jagarlamudi et al. (2010) improved the algorithm by proposing an improved initialization scheme and various other tricks to improve the robustness of KS solutions. They also consider application-specific details for an important domain of matching problems, natural language processing with specific tasks such as document alignment. Yamada and Sugiyama (2011) introduced another variant that replaces the HSIC measure with alternative kernel-based dependency measures, normalized cross-covariance operator and least-squares mutual information, using the same iterative learning algorithm as Quadrianto et al. (2010).

Recently, Djuric et al. (2012) provided an alternative optimization algorithm for kernelized sorting. Instead of directly optimizing the QAP, they relax the optimization problem by replacing the optimization space of permutation matrices with that of the doubly-stochastic matrices; positive matrices with unit row and column sums. The resulting cost is convex and hence they can find the global optimum of the relaxed cost. However, the solution does not in general correspond with the solution of the original cost function, and even though the algorithm does not need to solve assignment problems it is still computationally very demanding as it performs constrained gradient-based optimization over a N^2-dimensional parameter space. Also, while the model produces soft assignments between the samples, it does not correspond to a proper distribution over the permutations; it is merely the optimal solution over doubly-stochastic matrices. In the empirical sections, we will compare the proposed models primarily with CKS, since Djuric et al. (2012) showed that it typically outperforms other KS algorithms.

As mentioned already in the Introduction, the 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). In fact, the MCCA model is identical to our formulation (1) except that it does not have priors for any of the model parameters and it uses maximum likelihood estimation. The model by Tripathi et al. (2011) can

also be interpreted in similar fashion, since the maximum likelihood solution for probabilistic CCA is equivalent to the classical CCA solution. These two methods are hence the closest alternatives to ours, and in the empirical comparisons we will use them to demonstrate that the improved accuracy of the Bayesian solutions is because of the posterior inference. For running these comparisons we use the optimization algorithm as implemented by Tripathi et al. (2011) since it does not require initial seed pairing, and use the abbreviation CCA-ML to remind that the method corresponds to maximum likelihood estimation of a CCA-based matching method.

Another example of a method maximizing the join likelihood is the multilingual topic model by Boyd-Graber and Blei (2009). They learn a topic model for two languages by matching their vocabularies based on a maximum a posteriori estimation of a permutation matrix. While their eventual task is in learning the topic model itself, the matching solution is an integral part of the model.

Besides the above two criteria (maximal dependency and optimal joint model) for defining the right match, one can solve the matching problem also by explicitly constructing a distance between the two sets. This results in a non-iterative algorithm that merely needs to compute the distances once, since given the distances a single LAP solver will find the match. For example, Tripathi et al. (2011) used the manifold alignment method by Wang and Mahadevan (2009) to compute the distances by aligning local neighborhoods for the two sets in order to solve the matching problem. The application of Wang and Mahadevan (2009) also constitutes a good example of a potential application domain; they seek to match protein structures.

Some related work has also been done on posterior inference over permutations. While these works have not considered the matching problem as such, they are relevant background information. Kondor et al. (2007) considered exact variational inference over permutations by using Fourier transformations, and Plis et al. (2011) mapped the permutations to a high-dimensional hypersphere to do the same. These approaches are, however, only applicable to small N, at most tens, and hence would not be sufficient for our scenarios where N is in the order of hundreds. Leskovec et al. (2010), in turn, proposed a Metropolis sampler for permutations based on swaps of pairs. Their sampler is effectively equivalent to the one Gibbs-subset uses for sampling the posteriors, assuming we use $J = 2$; for larger J we can consider more complex operations than mere swaps of two pairs. They also provide illustrative characterizations of the properties of the permutation space, showing how only a tiny fraction of the permutations have non-negligible probability; this matches exactly our findings.

8 Method summary

Since the article describes two different inference algorithms and a number of variants for both, we will here summarize the previous sections by naming the different alternatives. After the summary, we provide the computational complexities for the variants and also for the closest competitors described in the previous section.

1. **Gibbs sampling with the most likely permutation (Gibbs-hard)**: Gibbs sampling for all other parameters except the permutation and the latent variables. The permutations are sampled with approximative Metropolis-Hastings step that jointly samples $\boldsymbol{\pi}$ and $\mathbf{Z}$ given the rest of the variables. The sampler is approximative since it accepts all proposals as if using Gibbs proposals, despite actually picking the most likely permutation given the latent variables and the rest of the parameters.

2. **Gibbs sampling with subset updates (Gibbs-subset)**: Gibbs sampling is used for all parameters. For sampling the permutation we select a random subset of J samples and draw the permutation corresponding to those samples from the true posterior, enumerating all possible permutations of the elements. In empirical experiments we used $J = 4$ and drew the values for 100 randomly chosen subsets for each posterior sample.
3. **VB with the most likely permutation (VB-hard)**: Variational approximation for all parameters, except the permutation. For the permutation, we simply use the most likely permutation $\hat{\boldsymbol{\pi}}$. This variant is equivalent to the comparison method CCA-ML, except that is does posterior inference over the CCA part instead of maximum likelihood.
4. **VB with local permutations (VB-local)**: Variational approximation for all parameters. 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.
5. **VB with numerical integration (VB-numInt)**: Variational approximation for all parameters. For estimating $q(\boldsymbol{\pi})$ we numerically integrate over $q(\mathbf{z}_i)$ to model the dependencies between the match and the latent variables. For each $\mathbf{Z}^{(m)}$ 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.

8.1 Computational cost

The computational complexity for one iteration of Gibbs-hard, VB-numInt and VB-hard is $\mathcal{O}(N^3 + N^2DK + NDK^3)$, where $D = \max(D_1, D_2)$. The first term is because of solving the LAP, the second for computing the costs of all possible pairs, and the last is for updating the parameters of the BCCA model. Here K is typically small compared to N and D. For VB-local the first term becomes $\mathcal{O}(N^4)$, since it needs to solve the LAP N times for each iteration. Gibbs-subset does not require solving LAP but instead it enumerates all permutations of size J, and hence the complexity is $\mathcal{O}(J! + N^2DK + NDK^3)$.

Even though the computational complexity is the same for most of the methods, the practical running times go up for the more accurate approximations. Gibbs-numInt needs to solve the LAP M times, and hence takes roughly M times longer than VB-hard since the LAP-step dominates the cost for all but very small N. To somewhat reduce the computational load, we update the match only after every 10 iterations; the permutations are anyway fairly stable. One iteration for Gibbs-hard is roughly as efficient as one iteration of VB-hard, but one typically needs to run the sampler for much longer than the VB algorithm that often converges in tens of iterations. For the practical experiments we used $1,000$ samples, making Gibbs-hard roughly as fast as VB-numInt and VB-local that require less iterations but solve LAP several times per iteration.

The computational complexity of the proposed methods is comparable to that of all of the competing methods. With the exception of CKS, all of the kernelized sorting methods and the CCA-based methods require repeatedly solving a LAP, which is the most time-consuming part in typical applications. Hence, each iteration takes the same amount of time as one iteration of our algorithms and the practical computation time depends on the number of iterations required for convergence. In practice, all methods are applicable to problems of similar magnitude, at least for hundreds of samples and possibly thousands with clever implementation. However, it is worth noting that for solving MCCA and CCA-ML one needs to invert a covariance matrix, introducing an additional complexity term of $\mathcal{O}(D^3)$; our models avoid this by modeling the correlations with explicit components.

CKS does not need to solve LAPs, but instead performs gradient-based optimization over a N^2-dimensional parameter space. There is no easy way to quantify the number of iterations needed for convergence, but computing the gradient as described by Djuric et al. (2012) is $\mathcal{O}(N^4)$ and hence at least for large N the iterations become slower than solving a LAP and the method is not applicable to as large problems as the competing methods. With the increased computational cost comes the advantage of guaranteed global optimum. In Sect. 9.4 we demonstrate how this advantage can be borrowed for the proposed methods by initializing the algorithms with the results of CKS.

9 Experiments

We start the experiment with an artificial data experiment, used to demonstrate the characteristics of the solution. In particular, we will show how the two alternative inference strategies have very different strength and weaknesses. We also demonstrate empirically the quality of the approximation for the Gibbs-hard sampler.

After the demonstration, we compare the proposed methods with earlier matching problem solutions, using data collections analyzed by the earlier authors. We perform three different comparisons. The first is an image matching task from Quadrianto et al. (2010), the second a metabolite matching task from Tripathi et al. (2011), and the last a cross-lingual document alignment task from Djuric et al. (2012). In all cases we compare the proposed methods with the leading kernelized sorting variant CKS by Djuric et al. (2012) and the CCA-ML method by Tripathi et al. (2011) which correspond to finding the maximum likelihood solution of our model. The purpose of these comparisons is to show that the proposed solution is more accurate than the earlier solutions, while also demonstrating that the improvement comes from the Bayesian treatment of the model.

9.1 Artificial data

In this section we will apply the model on simple artificial data sets of varying dimensionality, to illustrate an important property of the inference strategies. We generated data sets with $N = 40$ samples and D_x and D_y ranging from 10 to 640, by sampling data from the model (2) that has four latent variables. We then applied all model variants to these matching problems, initializing them with a permutation that has 50 % correct matches to simulate a reasonably good starting point. The resulting accuracies, averaged over 20 different data sets of each size, are shown in Fig. 2 (top), displaying an interesting trend: for low dimensionality the VB algorithms are the best, but for high dimensionality Gibbs-hard is clearly superior.

The reason for this in the shape of the posterior distribution over the permutations: for high-dimensional data it is peaked around the best permutation, whereas for low-dimensional data it is extremely wide; nearly all permutations are possible. This is illustrated in Fig. 2 (bottom), which shows for each of the dimensionalities the probability of the most likely permutation, computed for data with $N = 8$; for such a small set we can explicitly numerate all of the $40,320$ possible permutations and hence can compute the actual normalized probability. We see that when the dimensionality grows, the probability assigned for the most likely permutation gets larger. For low dimensionality, the posterior is very flat, but already for $D = 120$ the posterior is so peaked that the approximations of Gibbs-hard, VB-numInt and VB-local become accurate.

Next, we will explain why the variational approximation and the Gibbs sampler behave very differently for these two scenarios. Let us consider the high-dimensional case first. The

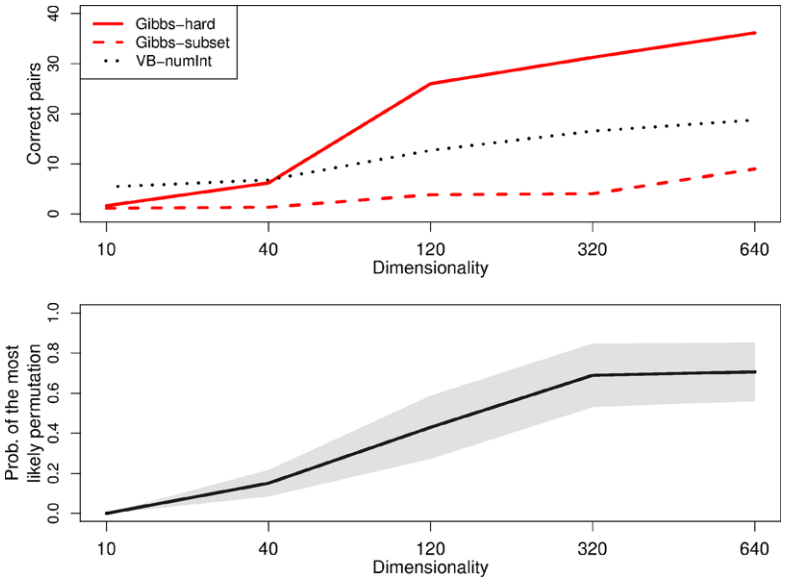

Fig. 2 *Top*: The matching accuracy of all methods increases with increasing data dimensionality (x-axis; note the non-linear axis), since there is more data for learning the match. Both Gibbs samplers are very bad for low dimensionality since the posterior over permutations is very wide, but for large dimensionality Gibbs-hard is clearly the best algorithm, reaching almost the perfect solution for this data with $N = 40$ samples. Gibbs-subset, however, is very inefficient in exploring the permutation space; a lot more iterations would be needed for reaching acceptable accuracy. The other two VB variants are roughly equivalent to VB-numInt on this data, and are hence not shown for clarity. *Bottom*: Illustration of how the posterior distribution over the permutations converges towards a delta distribution for increasing data dimensionality. The plot shows the probability of the most likely permutation $p(\hat{\boldsymbol{\pi}}|\boldsymbol{\Xi}, \text{rest})$ for a setup with $N = 8$, for which we can compute the full posterior exactly. Even though there are more than 40,000 possible permutations, the most likely one captures more than half of the posterior mass for the higher dimensionalities. The solid line depicts the mean probability, whereas the shaded area covers one standard deviation to both directions

Gibbs-hard does well because the assumption that $p(\boldsymbol{\pi}|\boldsymbol{\Xi}, \text{rest})$ corresponds to the most likely permutation is good, yet the sampler still explores the space of permutations effectively because $\boldsymbol{\Xi}$ is resampled every time. Gibbs-subset shows similar trend of improved accuracy for higher dimensions, but it is considerably less efficient in exploring the posterior since it does not find the best permutation for each sample but instead produces permutations with high degree of autocorrelation. This results also in clearly lower accuracy. The variational approximation, on the other hand, is a mean-field algorithm that explicitly averages over the permutations and the latent variables. Hence, for a peaked posterior the VB-local and VB-hard quickly get stuck with one permutation and the model converges to a local optimum. The VB-numInt model does better because it borrows the strength of the Gibbs-sampler; it numerically integrates over $\boldsymbol{\Xi}$ when updating the permutation, and hence can explore the space of permutations to some degree.

For the low-dimensional case the true posterior is very wide. The Gibbs-hard no longer approximates the posterior well, but even ignoring this issue we have a more fundamental problem with both samplers: Since the posterior over permutations is so wide, it becomes very difficult to estimate the rest of the parameters well. When the sampler, correctly, changes the permutation dramatically from one sample to another, it becomes nearly impossible for $\mathbf{W}$ and other parameters to converge towards reasonable posterior. The variational

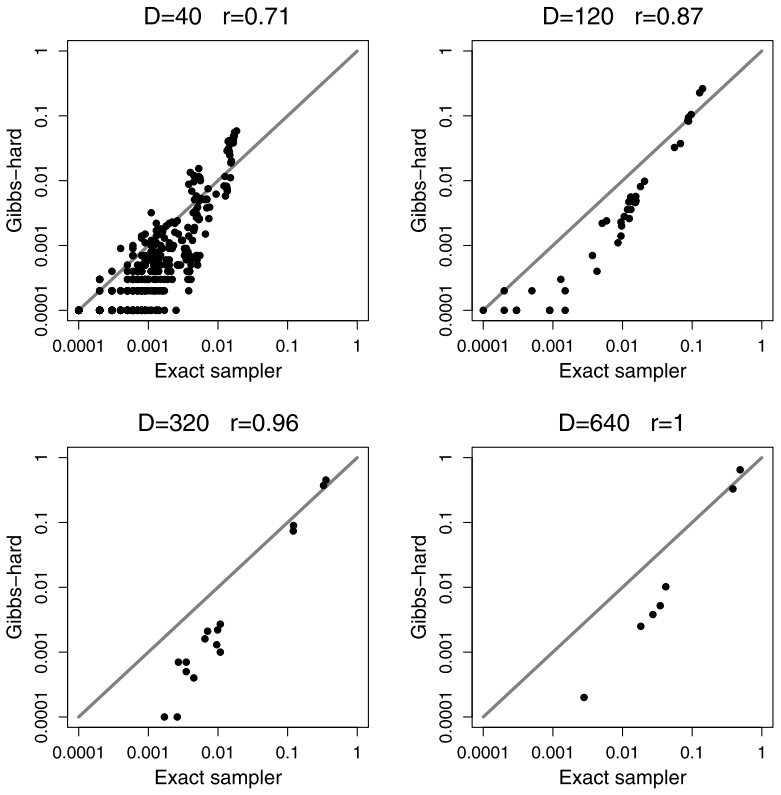

Fig. 3 The Gibbs-hard sampler is approximative since it approximates the conditional density $p(\boldsymbol{\pi}|\boldsymbol{\Xi}, \text{rest})$ with the most likely permutation. However, since the joint posterior $p(\boldsymbol{\pi}, \boldsymbol{\Xi}|\text{rest})$ depends more on $\boldsymbol{\Xi}$ than it does on $\boldsymbol{\pi}$, the algorithm still approximates the marginal posterior $p(\boldsymbol{\pi}|\text{rest})$ well. These log-probability crossplots compare the approximative posterior (y-axis) with the posterior obtained by running exact Gibbs sampler (x-axis), which is only feasible for very small sample sizes (here $N = 8$). The plot shows all permutations (individual dots) included at least once in either set of posterior samples. For exactly identical distributions all of the dots would lie along or close to the diagonal like. We see that the approximation, understandably, gives slightly too high probability (indicated by dots above the line) for the most likely permutation(s), but that it captures the general shape well and provides roughly the correct rank for the permutations. The Spearman rank correlation coefficient between the two sets of log-probabilities is always above 0.7 and for $D = 640$ the ranks are exactly correct

approximation, however, is inefficient in exploring this wide posterior since it averages over the possible values. Hence, the inference technique acts as a strong regularizer, making it possible to infer the rest of the parameters even though the true posterior over the matches is very wide.

Finally, we illustrate empirically the approximation error caused by the incorrect acceptance probability for the $(\boldsymbol{\pi}, \boldsymbol{\Xi})$ proposals in Gibbs-hard. Using a data set with $N = 8$ samples we ran both an exact Gibbs sampler and the proposed algorithm for $10,000$ independent samples and estimated the marginal posterior $p(\boldsymbol{\pi}|\text{rest})$ based on the posterior samples, keeping rest of the parameters except $\boldsymbol{\pi}$ and $\boldsymbol{\Xi}$ fixed. Figure 3 cross-plots the log-probabilities of the two distributions for various data dimensionalities. These plots suggest that despite making a seemingly crude approximation, the Gibbs-hard sampler still produces samples from almost the correct distribution, especially for high-dimensional data. It gives

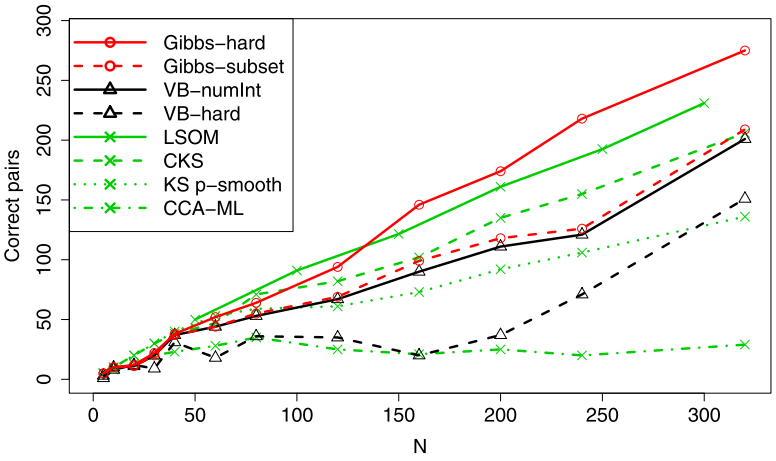

Fig. 4 Image matching: The number of correct matches as a function of the number of samples N. The Gibbs-hard variant is the best, outperforming the variational approximation variants, the maximum likelihood solution of the same model (CCA-ML), and also the state-of-art kernelized sorting methods convex kernelized sorting (CKS), least-squares object matching (LSOM), and p-smooth. The results for p-smooth are measured with a ruler from Jagarlamudi et al. (2010), the results for CKS are from Djuric et al. (2012), and the results for LSOM from Yamada and Sugiyama (2011) (who do not report the accuracy for $N = 320$). The VB-numInt and Gibbs-subset variants also perform well, outperforming p-smooth but not reaching the accuracy of CKS and LSOM. The color-coding (when available) and the symbols separate the Gibbs methods (*red*, *circle*) from the VB methods (*black*, *triangle*) and the comparison methods (*green*, *cross*)

somewhat too high probability for the most likely permutation, but it still gives non-zero probability mass for almost all of the same permutations as the exact sampler, and it also retains the relative probabilities of the permutations accurately.

9.2 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). The data has 2400 dimensions, and hence constitutes an example of a high-dimensional data for which the sampling algorithms should do well. VB-local, on the other hand, would not notably differ from VB-hard since the posterior is so peaked around the best permutation, and hence we leave it out from the comparison.

We solve the matching problem with varying subsets of the data. For VB-hard and VB-numInt, we learn $L = 50$ different initial models for each choice of N and initialize the final model by the consensus of these matches, using $K = 8$ components to keep the computational cost manageable. We then initialize the Gibbs variants with the result of VB-numInt, and use $K = 16$ components. We ran the samplers for 10 parallel chains, for 500 samples each, and then found the consensus of all posterior samples; since the initialization was already a good one we did not leave a burn-in period out. For Gibbs-subset we used $J = 4$ and 100 subset choices for each posterior sample.

Figure 4 compares the proposed methods with the p-smooth variant of kernelized sorting by Jagarlamudi et al. (2010), convex kernelized sorting by Djuric et al. (2012), and least-squares object matching (LSOM) by Yamada and Sugiyama (2011), all of which have been

demonstrated to be superior to the original kernelized sorting algorithm by Quadrianto et al. (2010). In addition, we compare the proposed methods with CCA-ML, which corresponds to using (hard) EM algorithm to find the maximum likelihood solution of our model. For CCA-ML we used an initialization strategy similar to what was used for the proposed methods. That is, we ran the model $L = 50$ times with different initializations that were slightly randomly permuted PCA-initializations. The final accuracy is the accuracy of the consensus; we also tried running the model one more time using the consensus as initialization but it typically decreased the accuracy. To avoid overfitting to the high-dimensional data, the CCA-ML method was ran on the first $N/8$ PCA components of each data set.

The main finding is that Gibbs-hard is the best matching solution for this data, followed by LSOM. For the whole collection with $N = 320$ images Gibbs-hard gets 275 correct matches compared to 136 for p-smooth and 206 for CKS;[1] Yamada and Sugiyama (2011) do not report an exact number for LSOM, but extrapolation suggests it would find roughly 245 correct pairs. The variational Bayesian inference is also good as long as we use numerical integration for estimating $q(\boldsymbol{\pi})$; it reaches accuracy comparable to CKS while outperforming p-smooth clearly for large sample sizes. The initialization scheme is necessary for achieving this; the individual runs used for finding the initialization only found on average less than 30 correct pairs for $N = 320$, whereas the final run initialized with their consensus reached 201.

Gibbs-subset, which was initialized with the output of VB-numInt, produces effectively the same results as its initialization; this confirms that the sampler is too inefficient in exploring the permutations space. Considerably more samples would be required to improve the results, but since Gibbs-hard works so much better we did not spend excess computational time to do this. We also see that the VB-hard variant that only uses the most likely solution is not sufficient here. This reveals that the good accuracy of Gibbs-hard and VB-numInt is because of the posterior inference over the permutations. However, for large N VB-hard still outperforms CCA-ML, demonstrating that Bayesian inference over the rest of the parameters already helps.

One of the advantages of the Bayesian matching solutions is that in addition to learning the best permutation we can characterize the posterior over the permutations. Convex kernelized sorting can also achieve this to some degree, since it optimizes the HSIC over doubly-stochastic matrices and hence produces soft assignments as a result. Next, we will compare how well the two methods fare in terms of such soft assignments. First we look at recall of the correct pairs, by ordering for each sample $\mathbf{x}_i$ the samples in $\mathbf{Y}$ according to the posterior probability of matching with $\mathbf{x}_i$. Figure 5 (left) shows how already 95 % of true pairs are captured within top 5 ranks. For comparison, Djuric et al. (2012) reports 81 % for the same threshold. We also inspected the actual probabilities to verify that the posterior is a reasonable distribution, and that they are consistent with the actual results. Figure 5 (right) plots the probabilities of the correct matches against the highest probabilities assigned for any pair. We see that the probabilities cover the whole range from roughly 0.1 to one, indicating that the algorithm is more certain of some pairs. We also note that it makes very few mistakes for the pairs that it assigned a high probability, indicating that the values indeed correspond to reasonable probabilities. For comparison, CKS does not assign a weight higher than 0.2 for any pair, illustrating how the soft match learned by CKS cannot be interpreted as any kind of probabilities even though they do sum up to one for each sample; the distributions are clearly too wide to represent the true uncertainty.

[1]Note that the subsets of images used for $N < 320$ may not be exactly the same that were used by the comparison methods, so only the case of $N = 320$ is directly comparable.

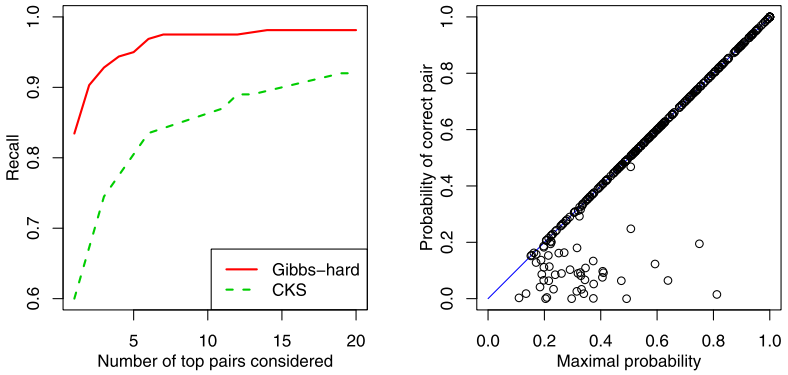

Fig. 5 *Left*: Recall of the correct matches when looking at the top I possible pairs in ranked order for each of the samples $\mathbf{x}_i$. The Gibbs-hard solution (*solid line*) clearly outperforms the convex kernelized sorting (*dashed line*) for all I. CKS requires top 5–6 matches to reach the same accuracy that Gibbs-hard captures already with the top-ranked pairs. *Right*: Cross-plot between the probabilities for the true pairs versus the highest probability assigned for any pair, for every sample $\mathbf{x}_i$. The samples lying on the *diagonal line* correspond to correctly identified pairs, cases where the highest probability is given for the correct pair. The *dots below the line* indicate mistakes. The important observation is that for the high-probability assignments almost all are correct; the *bottom right corner* of the image has only a few dots indicating mistakes when the model believed in some pair with high probability. The *bottom left corner* also shows correct behavior of more mistakes when the model is more uncertain; the values would not correspond to probabilities if the method got all of the matches correct for this regime as well

9.3 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 $\mathbf{X}$ and $\mathbf{Y}$ 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.

The individual time series are of very low dimensionality, ranging from 3 to 30 depending on the subject. Hence, we only apply the variational approximation methods for this problem; the posterior over the permutations is so wide that the Gibbs-sampler variants would not work at all. We then compare our method with CCA-ML and CKS.

Figure 6 shows how we again outperform the earlier methods. VB-numInt and VB-local have comparable accuracy, and both are better than CKS and CCA-ML. The comparison with CCA-ML, which corresponds to the maximum likelihood solution of the proposed model, confirms the findings of the image matching experiment (Sect. 9.2). The Bayesian solution is advantageous in two respects. First, the difference between VB-hard and CCA-ML comes solely from doing Bayesian inference over the CCA parameters, since these two models treat the permutations in identical fashion. More importantly, however, the difference between VB-hard and the other two variants reveals that already approximative Bayesian

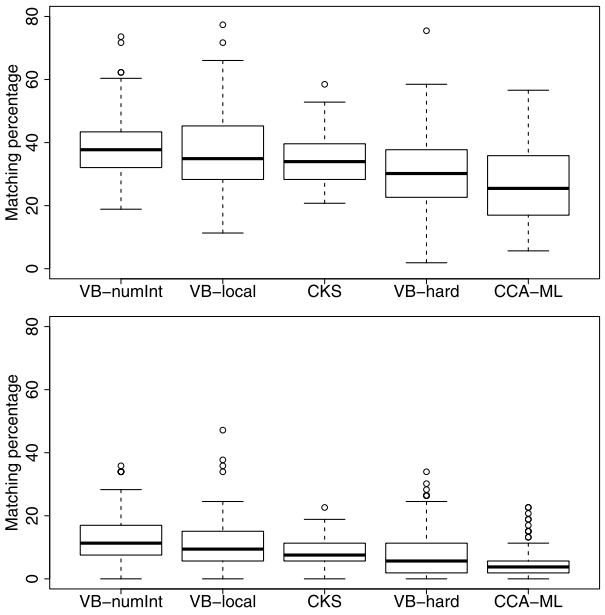

Fig. 6 Metabolite matching: The accuracy of matching the metabolites in two human populations, averaged over 100 different matching tasks and summarized as boxplots using the default parameters of the R implementation. The *top* plot summarizes the results for setup where the matches are constrained 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 proper variational approximations (both the numerical integration variant "VB-numInt" and the local perturbation variant "VB-local") are the best, followed by convex kernelized sorting (CKS). The difference between "VB-numInt" and "VB-local" is not statistically significant, but both are significantly better than the other three methods (paired t-test, $p < 0.01$)

inference over the permutations improves the accuracy dramatically. For completeness, we tried also the Gibbs samplers for this task, but as expected they did not work; they result in posteriors that are only marginally better than random assignments.

Note that in this experiment we did not use the advanced initialization strategy of learning the final model given a consensus of preliminary runs, but instead only used one initialization (based on the first PCA component) for each run. 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.

9.4 Document alignment

As a third real data experiment we consider the task of document alignment. Given two collections of documents written on two different languages, the task is to find the translations by matching the documents. We use the data provided by Djuric et al. (2012), consisting of more than 300 documents extracted from the Europarl corpus and represented as TF-IDF vectors of words stemmed with Snowball.[2] Djuric et al. (2012) considered nine different matching tasks, each between English documents and documents written in one of nine

[2] http://snowball.tartarus.org/.

Table 1 Matching accuracy (number of correct pairs) for multi-lingual document alignment tasks, where the goal is to pair the N documents with their English translations. Djuric et al. (2012) showed that the CKS method outperforms all other methods in this task, and hence we initialized the proposed methods with CKS to demonstrate how they can improve an already good solution. CKS already solves seven of the nine language pairs well, and the proposed methods are able to improve the accuracy also for the remaining two pairs (Finnish and Swedish). Gibbs-hard does particularly well, solving all cases with at least 98 % accuracy. VB-numInt and CCA-ML are also better than CKS for the two difficult language pairs, but do not reach perfect accuracy. The cases where the accuracy is below 95 % are written in boldface, to highlight cases that could not be considered solved with that method

Language	N	CKS	Gibbs-hard	VB-numInt	CCA-ML
Danish	387	385	385	379	385
Dutch	387	383	383	383	384
Finnish	308	**114**	308	**276**	**288**
French	356	356	356	356	356
German	356	356	354	356	352
Italian	387	385	381	383	384
Portuguese	356	356	356	356	354
Spanish	387	387	385	385	387
Swedish	337	**97**	337	**306**	**296**

other languages, and showed that CKS outperforms other kernelized sorting algorithms (the original KS algorithm, KS p-smooth and LSOM) for all tasks by a wide margin. They also achieved effectively perfect accuracy for seven of the tasks, reaching at least 98 % accuracy for each. For the remaining two language pairs, English-Swedish and English-Finnish, their accuracy was only 29 % and 37 %, respectively.

We initialized the Bayesian matching solutions with the permutation learned by CKS and then applied VB-numInt and Gibbs-hard for solving the same matching tasks, using a data representation that kept 10,000 words with the highest total TF-IDF weight over the corpus, separately for each language. For both methods we used $K = 16$, and for Gibbs-hard we again ran 10 separate chains for 500 samples each. We also applied CCA-ML with the same initialization, using $D_x = D_y = 50$ first PCA components for representing the data. The results are summarized in Table 1, showing how the Bayesian matching solutions and CCA-ML retain the good accuracy for the language pairs CKS already solved adequately. For the two difficult language pairs all methods improve on the initialization, but Gibbs-hard is the only one that solves also those problems perfectly, reaching 100 % accuracy.

9.5 Summary of the empirical experiments

Above we performed four separate experiment to evaluate the Bayesian matching solutions. Based on both the artificial and real matching experiments we can make the following conclusions:

– The proposed Bayesian matching solution outperforms the comparison methods, including kernelized sorting variants and earlier methods based on CCA. In particular, it is considerably more accurate than the maximum-likelihood solutions based on the same idea of introducing a permutation matrix as part of CCA (Haghighi et al. 2008; Tripathi et al. 2011). This confirms that the improved accuracy is because of the full posterior inference, instead of the model structure or cost function.

– For high-dimensional data Gibbs-hard is the best method. It can explore the posterior space more efficiently than the variational approximation, and it produces interpretable posterior estimates with high matching accuracy. While the conditional density used for sampling the permutation is not necessarily exact, the choice of always picking the best permutation is extremely efficient compared to more justified alternatives. As illustrated in Fig. 3, it is still very accurate in producing samples from the correct posterior.
– For low-dimensional data the true posterior over the permutations is so wide that properly modeling it does not produce good results. Hence, the Gibbs samplers do not work for such data. The variational approximations still provide accurate matches due to the inherent regularization effect of mean-field approximation.
– All of the proposed methods depend heavily on the initialization. A good initialization can be obtained by finding a consensus of several matches. Alternatively, the methods can be initialized by the result of the convex kernelized sorting method by Djuric et al. (2012); it finds the global optimum of a relaxation of the kernelized sorting problem and produces good matching accuracy.

The practical suggestion based on these observations is to use the Gibbs-hard method for learning the matching solutions, assuming the data dimensionality is sufficiently high (at least tens, preferably hundreds or more). The method should be initialized either with a consensus learned from multiple random initializations, or with the CKS method. The consensus is best learned with the VB variants, since the samplers might have difficulties with initial solutions where almost all pairs are incorrect; then the posterior is wide irrespective of the dimensionality since most BCCA components do not describe relationships between the two sets. For low-dimensional data, we suggest using the VB-numInt method instead of the samplers.

10 Conclusion

We introduced a variational Bayesian solution for the object matching problem introduced by Jebara (2004) and popularized by Haghighi et al. (2008), Quadrianto et al. (2010), Tripathi et al. (2011) and Yamada and Sugiyama (2011) for solving alignment tasks for example in natural language processing and computational biology. By learning together a Bayesian canonical correlation analysis model (Klami et al. 2013) and a permutation matrix re-ordering the samples in one of the sets, we obtained matching accuracies better than those of any earlier solution.

We presented two alternative inference strategies, one based on approximative Gibbs sampling and the other on variational approximation, and derived the computational details necessary for approximating the posterior over the permutations for both. The resulting algorithms were applied on three benchmark data sets and further illustrated on artificially generated data, to confirm that the proposed algorithms produce accurate matches. In particular, we outperformed all the earlier variants by a comfortable margin. For image matching we improved from 64 % to 85 %, for metabolite alignment we improved from 35 % to 39 %, and in two document alignment tasks we improved from 29–37 % to 100 %. These improvements correspond to real practical gains, and in particular the last one represents a qualitative change where the new method is able to perfectly solve a problem for which the earlier solutions were not satisfactory.

The Gibbs sampler was found to be the better of the two inference solutions, since it can more effectively explore the posterior space. We additionally showed that for sufficiently low-dimensional data the true posterior is so wide that it is actually better to concentrate

on some local region of the posterior space. For such setups the Gibbs sampler reduces to almost random guessing, and the variational inference is the best matching solution.

Acknowledgements The research was funded primarily by the TEKES, as part of the TIVIT Data to Intelligence (D2I) Program, and in part by Academy of Finland (Finnish Center of Excellence for Computational Inference COIN, 251170). We provide our grateful thanks for Prof. Matej Orešič for providing the data used in the metabolomics experiment, for Novi Quadrianto for providing the data for the image matching experiment, and for Nemanja Djuric for providing the code for CKS and the data for the document alignment task.

Appendix: Gibbs sampler details

The Gibbs sampler draws samples from the posterior $p(\mathbf{Z}, \boldsymbol{\pi}, \tau, \mathbf{h}, \mathbf{W}, \beta|\mathbf{X}, \mathbf{Y})$, by repeatedly sampling from the conditional densities summarized below. The equations are given for $\mathbf{x}$; the ones for $\mathbf{y}$ are obtained by replacing the subscripts. These equations are applicable also for Gibbs sampling of Bayesian CCA model without permutations, by simply setting $\boldsymbol{\pi} = \mathbf{I}$.

h The latent variables indicating the component activities are sampled independently, integrating $\mathbf{W}$ out in the process. Following Klami et al. (2013), this results in relative likelihoods

$$\frac{p(\mathbf{h}_{xk} = 1)}{p(\mathbf{h}_{xk} = 0)} = \frac{\gamma}{(1-\gamma)} \left(\frac{(\beta_{xk})^{-1}}{\lambda} \right)^{D_x/2} \exp\left(\frac{1}{2} \lambda \boldsymbol{\mu}^T \boldsymbol{\mu} \right),$$

where $\lambda = \tau_m \mathbf{Z}_{k:}^T \mathbf{Z}_{k:} + \beta_{xk}$ and $\boldsymbol{\mu} = \frac{\tau_x}{\lambda} (\mathbf{X} - \sum_{j \neq k} \mathbf{W}_{xj} \mathbf{Z}_{j:}^T) \mathbf{Z}_{k:}$. Here γ is the prior probability for $\mathbf{h}_{xk} = 1$, which we set to a constant $\gamma = 0.5$.

W The projections are sampled conditional on the spike-and-slab variable $\mathbf{h}$. For $\mathbf{h}_{xk} = 1$ we have $\mathbf{W}_{xk} \sim \mathrm{N}(\boldsymbol{\mu}, \lambda^{-1}\mathbf{I})$, where $\boldsymbol{\mu}$ and λ are as defined above, and for $\mathbf{h}_{xk} = 0$ we set $\mathbf{W}_{xk} = 0$.

β The prior precision for the active $\mathbf{W}_x$ has Gamma prior and hence also a Gamma posterior. For $\mathbf{h}_{xk} = 1$ we have

$$\beta_{xk} \sim \mathcal{G}\left(\alpha_0 + D_x/2, \beta_0 + \mathbf{W}_{xk}^T \mathbf{W}_{xk}/2\right),$$

and for $\mathbf{h}_{xk} = 0$ the value is drawn from the prior $\mathcal{G}(\alpha_0, \beta_0)$.

τ The noise precision has also Gamma prior, and the resulting posterior is

$$\tau_x \sim \mathcal{G}\left(\alpha_0^\tau + N D_x/2, \beta_0^\tau + \sum_{i=1}^{N} (\mathbf{x}_i - \mathbf{W}_x \mathbf{z}_i)^T (\mathbf{x}_i - \mathbf{W}_x \mathbf{z}_i) \right).$$

For τ_y we need to use the current permutation $\boldsymbol{\pi}$ to pick the latent variables, so that $\mathbf{W}_x \mathbf{z}_i$ is replaced by $\mathbf{W}_y \mathbf{z}_j$ such that $\boldsymbol{\pi}_{ji} = 1$.

$\mathbf{Z}, \boldsymbol{\pi}$ The latent variables and the permutation need to be sampled jointly, as described in Sect. 5. First we draw $\boldsymbol{\xi}_i \sim \mathrm{N}(0, \boldsymbol{\Sigma}_z)$ for each i independently, using $\boldsymbol{\Sigma}_z = (\tau_x \mathbf{W}_x^T \mathbf{W}_x + \tau_y \mathbf{W}_y^T \mathbf{W}_y + \mathbf{I})^{-1}$. Then we compute the most likely permutation given $\boldsymbol{\Xi}$ by solving the LAP with costs given by (4). Finally, we set $\mathbf{z}_i = \boldsymbol{\Sigma}_z \tau_x \mathbf{W}_x^T \mathbf{x}_i + \boldsymbol{\Sigma}_z \tau_y \mathbf{W}_y^T \mathbf{y}_j + \boldsymbol{\xi}_i$ according to the chosen permutation.

References

Andrieu, C., & Robers, G. O. (2009). The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, *37*(2), 697–725.

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

Boyd-Graber, J., & Blei, D. M. (2009). Multilingual topic models for unaligned text. In *Uncertainty in artificial intelligence*.

Burkard, R. E. (1984). Quadratic assignment problems. *European Journal of Operational Research*, *15*(3), 283–289.

Djuric, N., Grbovic, M., & Vucetic, S. (2012). Convex kernelized sorting. In *Proceedings of the 26th AAAI conference on artificial intelligence* (pp. 893–899).

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

Jagarlamudi, J., Juarez, S., & Daumé, H. III (2010). Kernelized sorting for natural language processing. In *Proceedings of the 24th AAAI conference on artificial intelligence (AAAI-10)* (pp. 1020–1025).

Jebara, T. (2004). Kernelized sorting, permutation, and alignment for minimal volume PCA. In *LNAI: Vol. 3120. Conference on computational learning theory (COLT)* (pp. 609–623).

Klami, A. (2012). Variational Bayesian matching. In *JMLR C&WP: Vol. 25. Proceedings of Asian conference on machine learning* (pp. 205–220).

Klami, A., & Kaski, S. (2007). Local dependent components. In *Proceedings of the 24th international conference on machine learning (ICML)* (pp. 425–432).

Klami, A., Virtanen, S., & Kaski, S. (2013). Bayesian canonical correlation analysis. *Journal of Machine Learning Research*, *14*, 899–937.

Knowles, D., & Ghahramani, Z. (2011). Nonparametric Bayesian sparse factor models with application to gene expression modeling. *Annals of Applied Statistics*, *5*(2B), 1534–1552.

Kondor, R., Howard, A., & Jebara, T. (2007). Multi-object tracking with representations of the symmetric group. In *Proceedings of the 11th international conference on artificial intelligence and statistics (AISTATS)*.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, *2*(1–2), 83–97.

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

Plis, S. M., McCracken, S., Lane, T., & Calhoun, V. D. (2011). Directional statistics on permutations. In *Proceedings of the 14th international conference on artificial intelligence and statistics (AISTATS)* (pp. 600–608).

Quadrianto, N., Song, L., & Smola, A. (2009). Kernelized sorting. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in neural information processing systems* (Vol. 21, pp. 1289–1296).

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

Smola, A. J., Gretton, A., Song, L., & Schölkopf, B. (2007). A Hilbert space embedding for distributions. In *LNCS: Vol. 4754. Algorithmic learning theory* (pp. 13–31).

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

Tripathi, A., Klami, A., & Kaski, S. (2009). Using dependencies to pair samples for multi-view learning. In *Proceedings of ICASSP 09, the international conference on acoustics, speech, and signal processing* (pp. 1561–1564).

Tripathi, A., Klami, A., & Virpioja, S. (2010). Bilingual sentence matching using kernel CCA. In *Proceedings of MLSP 2010, IEEE international workshop on machine learning for signal processing* (pp. 130–135).

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

Virtanen, S., Klami, A., & Kaski, S. (2011). Bayesian CCA via group sparsity. In *Proceedings of the 28th international conference on machine learning (ICML)* (pp. 457–464).

Yamada, M., & Sugiyama, M. (2011). Cross-domain object matching with model selection. In *Proceedings of the 14th international conference on artificial intelligence and statistics (AISTATS)* (pp. 807–815).

Wang, C., & Mahadevan, S. (2009). Manifold alignment without correspondence. In *Proceedings of the 21st international joint conference on artificial intelligence (IJCAI)* (pp. 1273–1278).