Leveraging Sequence Classification by Taxonomy-based Multitask Learning

Christian Widmer,[1] Jose Leiva,[1,2] Yasemin Altun,[2] and Gunnar Rätsch[1]

1 Friedrich Miescher Laboratory, Max Planck Society, Spemannstr. 39, 72076 Tübingen, Germany
2 Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany

Abstract. In this work we consider an inference task that biologists are very good at: deciphering biological processes by bringing together knowledge that has been obtained by experiments using various organisms, while respecting the differences and commonalities of these organisms. We look at this problem from an sequence analysis point of view, where we aim at solving the same classification task in different organisms. We investigate the challenge of combining information from related organisms, whereas we consider the relation between the organisms to be defined by a tree structure derived from their phylogeny. Multitask learning, a machine learning technique that recently received considerable attention, considers the problem of learning across tasks that are related to each other. We treat each organism as one task and present three novel multitask learning methods to handle situations in which the relationships among tasks can be described by a hierarchy. These algorithms are designed for large-scale applications and are therefore applicable to problems with a large number of training examples, which are frequently encountered in sequence analysis. We perform experimental analyses on synthetic data sets in order to illustrate the properties of our algorithms. Moreover, we consider a problem from genomic sequence analysis, namely splice site recognition, to illustrate the usefulness of our approach. We show that intelligently combining data from 15 eukaryotic organisms can indeed significantly improve the prediction performance compared to traditional learning approaches. On a broader perspective, we expect that algorithms like the ones presented in this work have the potential to complement and enrich the strategy of homology-based sequence analysis that are currently the quasi-standard in biological sequence analysis.

1 Introduction

Over a decade ago, an eight-year lasting collaborative effort resulted in the first completely sequenced genome of a multi-cellular organism, the free-living nematode *Caenorhabditis elegans*. Today, more than 50 eukaryotic genomes have been sequenced and several hundred more are underway. The genome sequences are the basis for much of the research on the molecular processes in these organisms. Typically, the more closely related the organisms are, the more similar are these processes. For some organisms, certain biochemical experiments for the analysis of particular processes can be performed more readily than for others (i.e. a large part of biological understanding was obtained from experiments based on a few model organisms such as yeast). This understanding can then be transferred to other organisms, for instance by verifying or refining models of the processes—often at a fraction of the original cost. This is but one example of a situation, where transfer of knowledge across organisms is very fruitful.

In computational biology we often study the problem of building statistical models from data in order to predict, analyze, and ultimately understand biological systems. Regardless of the problem at hand, be it the recognition of sequence signals such as splice sites, the prediction of protein-protein interactions, or the modeling of metabolic networks, we frequently have access to data sets for multiple organisms. Thus, our goal is to develop methods that aim at taking advantage of the data from different organisms in order to improve the performance of the statistical models built for all organisms. We argue that, when building a predictor for a given organism, data from other organisms should be incorporated to the extent of the relation between the organisms.

Since it is assumed that all life can be traced back to an ancient common ancestor, all organisms can ultimately be related by phylogeny. Furthermore, if two organisms share a sufficiently long evolutionary history before divergence, it can be expected that certain biological mechanisms (e.g., splicing) are conserved to some degree. Thus, it is reasonable to assume that we can leverage data from other organisms to enhance model quality for the organism of interest. In bioinformatics, this is traditionally done by considering sequence homology. This approach, however, is limited to almost exact correspondences of sequences between one or several biological sequences, while it fails to capture other features such as sequence composition that can be used to build an accurate model.

A family of machine learning methods, commonly referred to as *domain adaptation* or *transfer learning*, investigates the application of a predictor trained with data from a given domain to data from a different one (see e.g., [3, 5, 11]). Furthermore, the so-called *multitask learning* techniques consider the problem of simultaneously obtaining predictors from different domains by exploiting the fact that the domains are related (see e.g., [1, 6]). Most of these methods assume uniform relations across domains/tasks.[1] However, it is conceivable that from sharing information between closely related domains one can derive more benefit than from sharing between domains that are only distantly related (according to a given criterion). Hence, it is important to take into account the degree of relatedness among the domains when obtaining the set of models. Here, we investigate multitask learning scenarios where we are given *a priori* information about a hierarchy that relates the domains at hand, which is often the case for biological problems. In particular, we treat each organism as a domain and employ the hierarchy given by the phylogeny. The fact that the availability of data describing the same biological mechanism in several organisms is a reoccurring theme makes the hierarchical multitask learning approach particularly well suited for many applications in computational biology.

Building upon previous work [11], we propose a general framework for leveraging information from related organisms by ensuring correspondence on model basis rather than directly comparing sequences. We consider two principal approaches to incorporate relations across domains.

In the first approach, for a given task t, models from the other tasks serve as prior information to the model of t, such that the parameters $\mathbf{w}_t$ of task t are close to the parameters $\mathbf{w}_o$ of the other models. This can be achieved by minimizing the norm of the differences of the parameter vectors, $\|\mathbf{w}_t - \mathbf{w}_o\|$, along with the original loss function. A convenient way of implementing this approach is training models in a top-down manner, where a model is learned for each node in the hierarchy over the data sets of tasks spanned by the node. The parent nodes are used as the prior information, $\|\mathbf{w}_t - \mathbf{w}_{parent(t)}\|$. Here, one can readily use existing inference techniques with slight changes to the implementation. We describe this method in Section 2.2. Accordingly, one can use the models of all tasks as prior information, $\|\mathbf{w}_t - \mathbf{w}_{t'}\|$ for related tasks t and t', and train the parameters of all tasks jointly. This method is outlined in Section 2.3. Compared to the top-down approach, this formulation involves a larger set of parameters to be jointly optimized during training. However, it

[1] We use the terms *task* and *domain* interchangeably.

can be decomposed into sub-problems, which in turn are solved in an iterative manner. Its advantage is that each problem can be trained on smaller data sets compared to the top-down approach in which the model of the root node is trained on the union of all data sets.

An alternative to the latter pair-wise approach has been suggested in the context of support vector machines (SVMs) for the special case of two tasks [5]. We extend this idea and design an appropriate kernel function that not only considers the data of the task t, but also the data from all other tasks according to their similarities. We show that this kernel design can be derived by defining predictor functions over the task parameters as well as the parameters of the ancestors of the task. This leads to an essentially effortless multitask approach, since one can use standard SVM implementations by simply implementing the new kernel. We describe the new kernel and its derivation in Section 2.4.

In Section 3 we evaluate the proposed algorithms on simulated data and illustrate some of their properties. Moreover, we consider the problem of splice-site recognition in 15 different eukaryotic genomes and show that the proposed methods can significantly improve the prediction accuracy by combining the information available for all organisms. We conclude the paper with a discussion in Section 4.

2 Hierarchical Multitask Learning

2.1 Preliminaries

We are interested in the problem of learning M functions $f_t : \mathcal{X} \to \mathcal{Y}$ between the input space $\mathcal{X}$ and discrete output space $\mathcal{Y}$, where $t = 1, \ldots, M$ corresponds to a task. We assume that we are given the relations across tasks via a hierarchy $\mathcal{T}$, where the tasks are the leaves of $\mathcal{T}$. Our goal is to make use of the training sample S_t of task t, while also considering the training samples from the other tasks according to the given hierarchical information.

We investigate two principal approaches for exploiting hierarchical information about task relations in a multitask learning (MTL) framework; namely incorporating prior knowledge via *regularization* and via *kernel design*. The first approach is used in two of the proposed methods. Hence, we describe the underlying idea before we go into the technical detail. A regularization term is typically used to introduce a penalty for complex solutions into the optimization problem of a learning algorithm. In the existence of prior knowledge, the Empirical Risk Minimization

framework [13] incorporates the prior knowledge $\bar{f}$ by

$$\hat{f} = \min_{f} \left[R(f - \bar{f}) + \sum_{(\boldsymbol{x},y)\in S} \ell\left(f(\boldsymbol{x}), y\right) \right], \tag{1}$$

where R is the regularization term that penalizes the deviation of the current model f from the previously obtained (fix) model $\bar{f}$, and ℓ is a loss function (such as the squared loss, or the hinge loss) that penalizes the error on the training sample $S = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$. In the multitask approach, we use the models of related nodes as $\bar{f}$, where relations are given by $\mathcal{T}$. Following this scheme, any regularized Machine Learning framework (e.g., regularized least squares, regularized logistic regression) can be extended to include prior information.

Since one of the main goals of this work is to provide learning algorithms that scale to large amounts of data, we instantiate the concept for Support Vector Machines (SVM) [13].[2] It has been shown in previous work that SVMs using string kernels such as the Spectrum [8] or the Weighted Degree Kernel (WDK) [9] are well suited for nucleic and protein sequence analysis [2].

2.2 Top-Down Domain Adaptive Support Vector Machines

Our first approach uses a regularized multitask approach, where a model is learned for each node of the hierarchy $\mathcal{T}$ and the parent of node v serves as prior information to v, such that the final model of v is close to the model of its parent. The idea is to train the models in a top-down fashion, where the most general model is obtained at the root node and more domain specific models are obtained by moving down toward the leaves.

In SVMs, a model f is a linear function parametrized by $\mathbf{w}$ and b, $f = \langle \boldsymbol{x}, \mathbf{w} \rangle + b$. Since each model can be trained independently, we drop the indices for tasks and simply use $\mathbf{w}$ for parameters of the current node, and $\mathbf{w}_0$ for the parameters of the parent node. The primal of the extended SVM formulation is

$$\min_{\mathbf{w},b} \frac{1}{2} ||\mathbf{w} - \mathbf{w}_0||^2 + C \sum_{(\boldsymbol{x},y)\in S} \ell\left(\langle \boldsymbol{x}, \mathbf{w} \rangle + b, y\right), \tag{2}$$

where ℓ is the hinge loss, $\ell(z, y) = \max\{1 - yz, 0\}$. From a biological perspective, the penalty term $||\mathbf{w} - \mathbf{w}_0||^2$ enforces the model of an organism and the organism it is derived from to be similar, based on the

[2] Other options, however, may also be suitable to implement the ideas of this work.

assumption of a relatively small mutation rate. If the latter is the case, most properties of the model are conserved through evolution. In order to employ kernels, we derive the dual of the above formulation:

$$\max_{\alpha} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{n}\alpha_i \left(\underbrace{\left(\sum_{j=1}^{m}\alpha'_j y_i y'_j k(\mathbf{x}_i,\mathbf{x}'_j)\right)}_{p_i} - 1 \right), \tag{3}$$

$$\text{s.t. } \alpha^T\mathbf{y} = 0, \qquad 0 \le \alpha_i \le C \;\forall i \in \{1,n\},$$

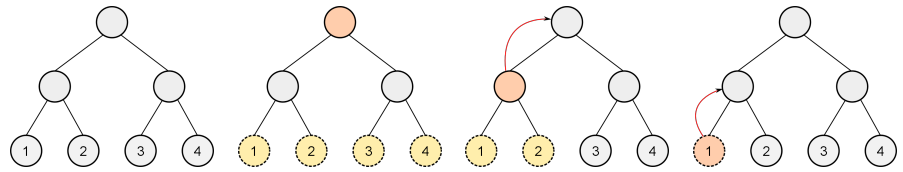
where n and m are the number of training examples in S and S_0 respectively, and S_0 is the training sample of the prior model. The α_i represent the dual variables of the current learning problem, whereas the α'_j represent the dual variables obtained from the prior model $\mathbf{w}_0$; in the case of the linear kernel, it is described as $\mathbf{w}_0 = \sum_{j=1}^{m}\alpha'_j y'_j \boldsymbol{x}'_j$. The resulting prediction is performed by

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}\alpha_i y_i k(\boldsymbol{x},\boldsymbol{x}_i) + \sum_{j=1}^{m}\alpha'_j y'_j k(\boldsymbol{x},\boldsymbol{x}_j) + b.$$

In the dual form of the standard SVM, the term $\mathbf{p}$ (corresponding to the p_i in Equation 3) is set to $\mathbf{p} = (-1,\ldots,-1)^T$. This is equivalent to the case, where we have no prior model (i.e. $\mathbf{w}_0 = (0,\ldots,0)^T$). In our extended formulation, the p_i can be pre-computed and passed to the underlying SVM-solver as the linear term of the corresponding quadratic program (QP). To provide implementations that readily deal with large-scale learning problems, we have extended the SVM implementations *LibSVM* [4] and *SVMLight* [7] to handle prior information.[3]

Top-Down Hierarchical Learning An illustration of the training procedure is given in Figure 1. In a top-down manner, a predictor f is obtained at each node v, whereas the loss L is evaluated on the union of training data $S = \cup\{S_{t \preccurlyeq_{\mathcal{T}} v}\}$ at the leaves below the current node v, while the current model f is regularized against the parent predictor f_p. Training is completed once we have obtained a predictor f_t for each task t. The algorithm will be referred to as *Top-Down-SVM* in the following.

[3] http://www.shogun-toolbox.org



(a) Given taxonomy (b) Root training (c) Inner training (d) Taxon training

Fig. 1. Illustration of the hierarchical top-down MTL training procedure. In this example, we consider four tasks related via the tree structure in 1(a). Each leaf is associated with a task t and its training sample $S_t = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$. In 1(b), training begins by obtaining the predictor at the root node, for which data from all leaves are taken into account in the loss term. Next, we move down one level to train a classifier at an inner node, as shown in 1(c). Here, the loss is measured w.r.t. $S_1 \cup S_2$ and the classifier is forced to be similar to the parent solution via the regularization term, as indicated by the red arrow. Finally, in 1(d), we obtain the final classifier of task 1 by only taking into account S_1 to measure the loss, while again regularizing against the parent predictor. The procedure is applied in a top-down manner to the remaining nodes until a predictor for each leaf was obtained.

2.3 Support Vector Machines with Pairwise Task Regularization

In the previous section, we described a method that learns models for internal nodes of the hierarchy and imposes regularization between a node and its parent. In this section, we describe a method where the relation between tasks is modeled directly via pairwise regularization. We refer to this method as *Pairwise* learning. Similar regularization-based multitask learning methods were proposed in [6]. However, we extend this previous approach by incorporating hierarchical task information, and by providing an efficient decomposition for practicability.

Given the hierarchy $\mathcal{T}$, we first compute the distance $d_{s,t}$ between the tasks s and t by summing up the lengths of the edges on the path from one leaf to the other (tree-hop-distance). In our experiments, we assumed all edge lengths to be 1. The resulting distance $d_{s,t}$ is subsequently converted to a similarity $\gamma_{s,t}$ by the transformation $\gamma_{s,t} = a - d_{s,t}/d_{max}$, where $a \geq 1$ is a parameter to control the base similarity between tasks, and d_{max} is the maximal distance between any pair of leaves in $\mathcal{T}$. The matrix Γ that captures all pairwise similarities $\gamma_{s,t}$ will be referred to as task similarity matrix in the following.

In this method, the *prior* information comes from the models of similar tasks. All the models are trained jointly by optimizing the regularization term along with the loss ℓ, which is measured separately for each $\mathbf{w}_t$ on

the corresponding data set S_t,

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_M} \quad \frac{1}{2}\sum_{t=1}^{M}\sum_{s=1}^{M}\gamma_{s,t}\|\mathbf{w}_t-\mathbf{w}_s\|^2+\sum_{t=1}^{M}C_t\sum_{(\boldsymbol{x},y)\in S_t}\ell\left(\langle\boldsymbol{x},\mathbf{w}_t\rangle,y\right).$$

The parameter C_t is used to trade off the loss for task t on the training sample against the regularization term in order to control generalization performance. In our experiments, we set $C_t = C$, for $t = 1,\ldots,M$ for simplicity. The biological interpretation of this formulation is that if two organisms share a sufficiently long evolutionary history before divergence (reflected by $\gamma_{s,t}$), it can be expected that certain aspects of the model describing the biological mechanism are conserved. Thus, we use the task-similarity matrix $\boldsymbol{\Gamma}$ to control how strongly we regularize each $(\mathbf{w}_t,\mathbf{w}_s)$ pair to be close to each other.

Decomposition The pairwise formulation learns models only for the leaf nodes of the hierarchy, as opposed to the *Top-Down-SVM* approach, in which the number of models to learn is given by all the nodes of the hierarchy. Its comparative disadvantage, on the other hand, is that it couples the parameters of all models, which leads to a large optimization problem. In order to overcome this limitation, we developed a decomposition of the optimization problem that allows the global solution to be obtained by solving a series of SVM-like quadratic programs iteratively until convergence. It can be shown that the above optimization problem has a fixed point that coincides with the optimization problem of

$$\min_{\mathbf{w}_t}\ \tfrac{1}{2}\lambda_t\|\mathbf{w}_t-\boldsymbol{r}_t\|^2+C\textstyle\sum_{(\boldsymbol{x},y)\in S_t}\ell\left(\langle\boldsymbol{x},\mathbf{w}_t\rangle,y\right),\quad\forall t \tag{4}$$

$$\boldsymbol{r}_t=\textstyle\sum_{s\neq t}\beta_{ts}\mathbf{w}_s, \tag{5}$$

$$\lambda_t=\textstyle\sum_{s\neq t}\gamma_{ts},\quad \beta_{ts}=\gamma_{ts}/\lambda_t.$$

This formulation decouples the optimization problem into individual tasks and, hence, retains scalability. It states that the regularization prior $\boldsymbol{r}_t$ of each task should be in the convex hull of $\{\mathbf{w}_s\}_{s\neq t}$. It can be solved iteratively by finding the optimal $\boldsymbol{r}_t$ for the current $\mathbf{w}_t$ (cf. (5)) and finding the optimal $\mathbf{w}_t$ for the current $\boldsymbol{r}_t$ (cf. (4)) for each task until convergence. Note that the difference between (4) and (2) is the prior model, where in the first case it is the weight average of related tasks and in the latter it is the parent model. Hence, the SVM implementations in Section 2.2 can be used to solve (4). Kernelization follows similarly. Investigating the

relation between the dual and primal parameters,

$$\mathbf{w}_t = \sum_{i:\boldsymbol{x}_i \in S_t} \alpha_i y_i \boldsymbol{x}_i + \sum_{s \neq t} \beta_{ts} \sum_{j:\boldsymbol{x}_j \in S_s} \alpha_j y_j \boldsymbol{x}_j,$$

reveals that the pairwise regularization method results in "borrowing" support vectors of related models with respect to the task similarities.

2.4 Multitask-Kernel Learning

In Sections 2.2 and 2.3, we presented two hierarchical multitask approaches based on regularization with respect to related models. In this section, we propose an alternative approach, *MultiKernel*, by defining a kernel function that incorporates data from related tasks. We first define a matrix Λ to encode ancestry relationships as follows: Let λ_{tr} be the similarity of a task t and an inner node r. As in the previous section, λ_{tr} is inversely related to the distance between t and r, if r is an ancestor of t with respect to $\mathcal{T}$, $t \preccurlyeq_{\mathcal{T}} r$, and 0 otherwise. We then define the predictor function for task t as

$$f_t(\boldsymbol{x}) = \mathbf{w}_t^T \boldsymbol{x} + b_t = (\boldsymbol{u}_t + \sum_{t \preccurlyeq r} \lambda_{tr} \boldsymbol{v}_r)^T \boldsymbol{x} + b_t,$$

where $\{\boldsymbol{v}\}_r$ are the internal node parameters and $\boldsymbol{u}_t$ are the leaf node (task) parameters. Here, the parameters of the node at each level in the hierarchy represent the corresponding level of abstraction. More precisely, the parameters of the root node capture the most common structure across all tasks and as we descend in the hierarchy, the parameters capture the deviation from the previous level. Our goal is to achieve the proper specialization level for each task by combining these parameters in the prediction function.

We propose to obtain $\{\boldsymbol{u}_t\}$ and $\{\boldsymbol{v}_r\}$ by solving the regularized loss function for all tasks,

$$\min_{\{\boldsymbol{u}_t\},\{\boldsymbol{v}_r\}} \frac{1}{2} \sum_{t=1}^{M} \|\boldsymbol{u}_t\|^2 + \frac{1}{2} \sum_{r=1}^{R} \|\boldsymbol{v}_r\|^2 + C \sum_{t=1}^{M} \sum_{(\boldsymbol{x},y) \in S_t} \ell\left(\langle \boldsymbol{x}, \mathbf{w}_t \rangle, y\right),$$

where ℓ is the hinge loss, and R is the number of inner nodes. Note that the tasks are related to each other through the internal node parameters $\boldsymbol{v}$ as in the case of *Top-Down-SVM*. However, the loss term ℓ is evaluated only on the leaf nodes, as opposed to *Top-Down-SVM*, where ℓ is evaluated at all nodes in the hierarchy by combining the relevant data

sets. Instead of learning the internal node models by error minimization and then enforcing the models of the parent-child nodes to be similar, the goal here is to learn the internal node models directly by minimizing the error on the leaf nodes (i.e. the tasks). It can be shown that the dual formulation of the problem above is equivalent to that of the standard SVM

$$\max_{\alpha} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \tilde{k}(\mathbf{x}_i,\mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$
$$\text{s.t.} \quad \alpha^T\mathbf{y} = 0, \quad 0 \le \alpha_i \le C \ \forall i \in \{1,n\},$$

where the kernel is defined on the union of all data sets,

$$\tilde{k}(\boldsymbol{x}_i,\boldsymbol{x}_j) = \tilde{\gamma}_{t(i),t(j)}k(\boldsymbol{x}_i,\boldsymbol{x}_j).$$

Here, $t(i)$ denotes the task that data point $\boldsymbol{x}_i$ belongs to, and $\tilde{\gamma}_{ts}$ are the entries of $\tilde{\boldsymbol{\Gamma}} = \mathbf{I} + \Delta\Delta^T$. We require $\tilde{\boldsymbol{\Gamma}}$ to be positive semi-definite in order to yield a valid kernel $\tilde{k}$. Hence, we have constructed a kernel $\tilde{k}$ that incorporates the interaction among tasks in addition to the interaction among data points. This formulation is a generalization of the domain adaptation method of [5], where the reweighting of the original kernel matrix is less flexible. Lastly, please note that, while we needed Δ for the sake of derivation, it becomes clear from the dual formulation that $\tilde{\Gamma}$ may be obtained using the same transformation as described in Section 2.3, which was done for the following experiments.

3 Results

Experimental Setup We performed experiments on two types of data sets. First, we considered synthetic sequence data, which was created by applying mutations to a Position Specific Scoring Matrix (PSSM) according to a pre-defined binary tree structure. Balanced, equally sized training data sets, 100 examples each, were sampled for each of the leaves. As test set, an additional 5000 examples were sampled for each task. We used the area under the ROC curve to evaluate the prediction performance. (An evaluation using the area under the precision recall curve yields similar results.)

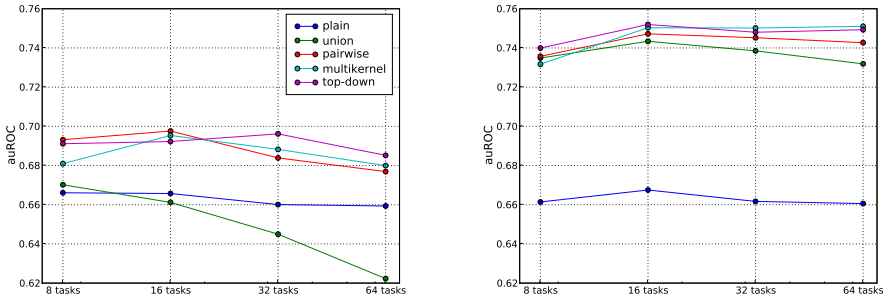
As a second application, we considered the problem of splice site recognition. We generated and used labeled sequences of acceptor splice sites from 15 different eukaryotic genomes, which were similarly generated in [10, 11] (see Figure 3 for the taxonomic relation between the organisms).

For each task, we obtained 10000 training examples and an additional test set of 5000 examples. We normalized the data sets such that there are 100 negative examples per positive example. We report the area under the precision recall curve (auPRC), which is an appropriate measure for unbalanced classification problems (i.e. detection problems) [9].

Experiments were performed for each of the presented MTL methods (*Top-Down-SVM*, *Pairwise*, *MultiKernel*) and the following two baseline methods. In *Union*, all data are combined into one data set $S = \cup_{t=1}^{M} S_t$, and a single global model is obtained, which is used to predict on all domains. On the other extreme, we consider the baseline method *Plain*, in which an individual SVM is trained on the data S_i of each domain separately, not taking into account any information from the other domains. For each method, the regularization parameter C was chosen by cross-validation with 4 and 3 splits for toy and splice data, respectively. After obtaining the optimal C, we retrained the model on all available training data. The performance was measured on the separate test sets, which are considered large enough to obtain reliable estimates of the predictors' performances. The data sets, as well as the Appendix with more detail about hyper-parameter selection and toy data generation will be made available on our web-site[4].

Results on the Toy Datasets We consider two different settings of generating the toy data sets: one with relatively high mutation rate (larger differences between neighbouring tasks in the hierarchy) and one with small mutation rate (all tasks are closely related). For this study, we analyze how the different methods perform in these two settings with respect to the number of tasks (8, 16, 32, 64 tasks). The results are shown in Figure 2. We can observe that the two baseline methods perform very differently: While the *Plain* method performs essentially indifferent for different numbers of tasks and mutation rates, the *Union* method performs very well when the mutation rate is low, but quite poorly for large mutation rates. Moreover, the performance of *Union* degrades for an increasing number and, hence, diversity of tasks. This is particularly pronounced for high mutation rates. The three proposed methods all perform better than the two baseline methods, indicating that it pays-off to take the additional data and the relation between the tasks into account. However, among the three proposed methods there is no method that consistently performs best.

[4] http://fml.mpg.de/raetsch/suppl/mtl-taxonomy



(a) Toy data: High mutation rate

(b) Toy data: Low mutation rate

Fig. 2. Results for the five considered methods on the toy data sets with high and low mutation rate. Shown are the average areas under the ROC curves for different numbers of tasks that have been generated from full binary trees.

Results on the Splice Dataset On the toy data set, all three MTL methods perform similarly well and clearly outperform the two baselines on all eight tasks. For the splice site data set, examining the average performance across all organisms (the last column in Figure 4), we also observe superior performance of the MTL methods over the baseline methods. The *Plain* method is again outperformed by all other methods, which emphasizes the importance of using data from related organisms. Comparing the hierarchical MTL methods, we observe that the *Top-Down-SVM* approach performs slightly better than the other MTL methods. We conjecture that this is due to a suboptimal choice of the similarity matrix, where tree-hop-distance was used for the pairwise and multikernel approach.

Zooming in on the results for individual organisms, we observe the same trend: MTL methods outperform the baselines in most cases and the *Plain* method yields the worst performance. For 11 out of 15 organisms, all MTL methods perform better than the baseline. The *Top-Down* method always performs better than the baseline methods, except for *A. nidulans*, which is the only fungal organism in our set. The gain from hierarchy is more pronounced in the lower levels of the hierarchy, e.g., for *A. thaliana*, *O. sativa*, *O. latipes*, and *D. rerio*, where data from closely related organisms are used to leverage the learning process. An exception to this behaviour is seen on the mammals branch, where the performance gain from MTL methods gets smaller for *H. sapiens* and essentially diminishes for *M. musculus*. Our conjecture is that the hierarchy for this

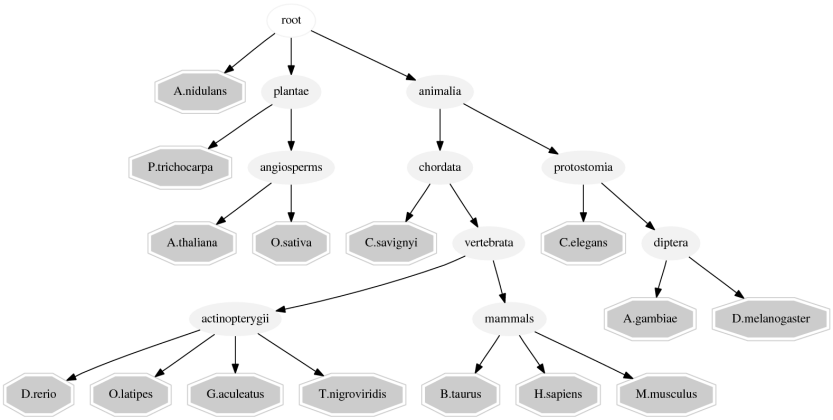


Fig. 3. Taxonomy used to relate organisms for the splicing experiment

branch is not deep enough in order to represent closely related organisms. Including more similar organisms, and hence, extending the hierarchy to capture more evolutionary steps, can improve the performance gain for these organisms.

It is worthwhile to investigate the performance of the methods on *A. nidulans*, which is the child of the root node and, therefore, most distantly related to the other organisms. We observe that the *Union* method performs worst on this organism and the MTL methods perform on the same level as the *Plain* method. Hence, MTL methods manage to improve the performance for closely related organisms, while causing (essentially) no performance loss on the distantly related ones.

Moreover, it is interesting to observe that *A. nidulans* and *C. elegans* are the only organisms/tasks, for which the *Union* method is considerably worse than the *Plain* method. This hints at major differences between the recognition of splice sites in these two organisms. This is less surprising for *A. nidulans* as it is for *C. elegans*. It appears worth investigating this property also for other nematode genomes to better understand this observation.

4 Discussion and Conclusion

We outlined two principle ways of leveraging information from related organisms, where relationship between organisms is defined with respect to a given hierarchy. We presented three algorithms that readily deal with large scale problems such as those frequently encountered in genomic

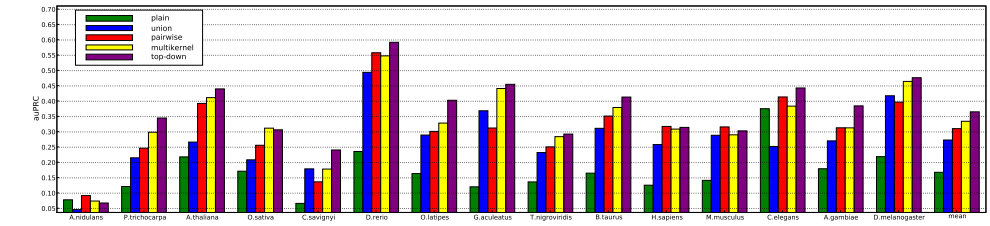


Fig. 4. Results for the splice site data sets from 15 eukaryotic genomes: Shown are auPRC performances of the five considered methods for each organism (two baseline methods: *Plain*, *Union*; three proposed methods: *Top-Down*, *Pairwise*, and *Multikernel*). We can observe that the two of the MTL methods consistently outperform the baseline methods. In 14/15 cases, the hierarchy information lead to improved prediction results.

sequence analysis. We have demonstrated that our methods outperform baseline methods on synthetic data, and data from splice site prediction. On the one hand, the poor performance of *Plain* relative to the MTL methods shows that exploiting information from other tasks is in fact beneficial. On the other hand, the poor result of *Union* demonstrates that there is no single model that fits all tasks equally. Clearly, methods that carefully combine the data from different tasks according to their relatedness perform best.

We are encouraged by the good performance of the *Top-Down-SVM* method, as it provides a fast, simple and non-parametric way of exploiting hierarchical information. Inferring an accurate task-similarity matrix $\boldsymbol{\Gamma}$ proves to be non-trivial, therefore one should think of additional ways of using the hierarchy to facilitate that task. Experiments on the splicing data show that a simple task-similarity matrix based on tree-hop-distance can be suboptimal, particularly for cases when edge lengths (i.e. evolutionary distance to the parent) are unequal. Our immediate future work involves experiments, where edge lengths are incorporated into the similarity matrix. For phylogenetic trees, edge lengths can, for instance, be given by evolutionary years.

From our experience with Multitask learning experiments, we conclude that certain requirements have to be fulfilled in order for MTL method to be beneficial. In particular, the problem has to be difficult enough to require considerable amounts of data. In this context, difficult means that the number of training examples for each task is relatively low, compared to the complexity of the model (e.g., we need many training examples to learn good models, when using string kernels of high degree).

Furthermore, the tasks have to be similar enough to contain mutually relevant information. If tasks are too different, and the learning problem is reasonably easy, we might be better off learning tasks independently. On the contrary, if task are too similar MTL methods will not give rise to much improvement over obtaining one global model (as shown in the toy-data experiment).

Assuming reasonable conditions in terms of problem difficulty and task similarity, we can benefit most from a hierarchy if we consider relatively many tasks. Here, we need the fine-grained information about task relations contained in a hierarchy to direct the trade-off in our learning approaches. In particular, we can benefit most compared to one global model (i.e. *Union*), if the hierarchy describes a rich structure (e.g., not a trivial one, such as all leaves attached to the root).

For all of the presented methods, we plan to provide publicly available scalable implementations based on modified versions of *SVMLight* [4] and *LibSVM* [7] as part of the Shogun Machine Learning Toolbox [12]. Lastly, we would like to emphasize that in computational biology there is a great number of problems for which corresponding data sets are available for multiple organisms. We expect that hierarchical MTL methods can indeed make a big difference for this class of problems. Therefore, the methods and implementations that we presented could be of value to a wide range of applications.

Acknowledgments We would like to thank Sören Sonnenburg for help with the implementation of the presented algorithms. Also, we acknowledge Cheng Soon Ong for providing the raw splicing data sets. Furthermore, we would like to thank Gabriele Schweikert, Georg Zeller and Klaus-Robert Müller for inspiring discussions. This work was supported by the DFG grant RA1894/1-1.

References

1. Rie K. Ando and Zhang, T. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, November 2005.
2. A. Ben-Hur, C.S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch. Support vector machines and kernels for computational biology. *PLoS Comput Biol*, 4(10):e1000173, Oct 2008.
3. J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. *Advances in Neural Information Processing Systems*, 20:129136, 2008.
4. C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines, 2001.
5. H. Daumé. Frustratingly easy domain adaptation. In *ACL*. The Association for Computer Linguistics, 2007.
6. T. Evgeniou, C.A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.
7. T. Joachims. SVMLight: Support Vector Machine. *SVM-Light Support Vector Machine http://svmlight. joachims. org/, University of Dortmund*, 1999.
8. C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575, 2002.
9. G. Rätsch and S. Sonnenburg. *Accurate Splice Site Detection for Caenorhabditis elegans*. MIT Press, 2004.
10. G. Rätsch, S. Sonnenburg, J. Srinivasan, H. Witte, K.-R. Müller, R. Sommer, and B . Schölkopf. Improving the *C. elegans* genome annotation using machine learning. *PLoS Computational Biology*, 3(2):e20, 2007.
11. G. Schweikert, C. Widmer, B. Schölkopf, and G. Rätsch. An empirical analysis of domain adaptation algorithms. In *Advances in Neural Information Processing System, NIPS*, volume 22, Vancouver, B.C., 2008.
12. S. Sonnenburg, G. Rätsch, S. Henschel, C. Widmer, A. Zien, F. de Bona, C. Gehl, A. Binder, and V. Franc. The shogun machine learning toolbox (under revision). *Journal of Machine Learning Research*, 2010.
13. V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, New York, 1995.