
Scalable Hierarchical Multitask Learning in Sequence Biology

Anonymous Author(s)

Affiliation

Address

email

Abstract

Multitask learning methods investigate the challenge of combining information from several related problem domains. For a large family of multitask problems, relationships between tasks can be described by a hierarchical structure. This is particularly the case for many problems in Computational Biology, where different tasks correspond to different organisms, whose relationship to each other is defined by phylogeny. In this work we describe a set of algorithms that combine large-scale classification methods with ideas from Domain Adaptation and Multitask Learning. We introduce several methods of exploiting hierarchical task relations for Multitask Learning. These algorithms are designed for large-scale applications and scale to problems with a great number of training examples. The performance of the presented methods is demonstrated on synthetic data, as well as on splice-site data arising from a problem in genomic sequence analysis.

1 Introduction

In Machine Learning, model quality is most often limited by the lack of sufficient training data. In the presence of data from several related domains, the goal is to boost the performance by making use of all available information. When building a model that incorporates information from several domains, it is important to take into account the degree of relatedness among the domains. Clearly, sharing information from closely related domains can bring more benefit than the domains that are only distantly related. However, most Multitask learning methods have considered uniform relations across domains. In this paper, 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 Computational Biology, a common task is to build a statistical model 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 datasets for multiple organisms. The fact that the availability of data describing the same biological mechanism in several organisms is a reoccurring theme makes hierarchical multitask methods particularly well suited for applications in Computational Biology.

Since all life can be traced back to an ancient common ancestor, all organisms can ultimately be related by phylogeny. Furthermore, if two organisms shared 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. The problem with that approach is that it is limited to (almost) exact correspondences of sequence motifs 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.

Building upon previous work [6], we therefore propose a general framework of leveraging information from related organisms, by ensuring correspondence on model basis, rather than directly comparing sequences. In particular, we present two principal ways of incorporating hierarchical relations into Multitask learning. The first set of approaches involve extending standard learning methods via designing regularization terms and kernel methods using the hierarchy. The second approach involves designing a novel optimization problem tailored for the task in hand. Experimental results on simulated data and splice-data data for biological sequence analysis shows the validity of our approaches.

2 Methods

There exist two principal approaches of exploiting hierarchical information about task relations in a Multitask Learning framework. One possibility is to first infer a task similarity matrix γ from the hierarchy \mathcal{T} and then to employ the inferred task similarity in regular Multitask Learning algorithms. In the simplest case, the mapping from \mathcal{T} to γ could be as simple as counting the number of edges that separate two tasks (or equivalently summing over the edge weights, if present) and then converting this distance to a similarity. Although the design of the mapping is a central component, it is not the main focus of this work. We therefore do not elaborate this in detail due to space constraints, but refer the reader to the final publication which is in preparation.

The second approach to incorporate hierarchical information is to design an algorithm that exploits the given taxonomy directly. In the proposed algorithm, a model is learned at each node of the hierarchy in a top-down fashion, where the most general model is obtained at the root node and more domain specific models are obtained as we move down towards the leaves. The main idea is that the model at the parent node always serves as prior information to the model of the current node, such that the final model at a given node v is close to the model of the parent node of v via regularization.

2.1 Regularized Domain Adaptation

To realize these concepts, we first introduce a general technique of incorporating prior information into the learning algorithm via the regularization term, as previously proposed in [3]. The regularization term is typically used to introduce a penalty for complex solutions into the optimization problem of a learning algorithm.

Generally, in the Empirical Risk Minimization framework, the incorporation of prior knowledge \bar{f} can be expressed as

$$\hat{f} = \min_f \left[R(f - \bar{f}) + \sum_{(\mathbf{x}, y) \in S} L(y - f(\mathbf{x})) \right],$$

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 $S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$. Following this scheme, any regularized Machine Learning framework (e.g. regularized least squares, regularized logistic regression) could be extended to include prior information.

2.1.1 Domain Adaptive Support Vector Machine

As one of the main aims of this work is to provide learning algorithms that readily scale to large amounts of data, we instantiate the above concept for the Support Vector Machine (SVM). It has been shown in previous work that the SVM using string kernels such as the Spectrum [4] or the Weighted Degree Kernel (WDK) [5], is well suited for nucleic and protein sequence analysis [1].

The primal of the extended SVM formulation is given by

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

where ℓ is the hinge loss, $\ell(z, y) = \max\{1 - yz, 0\}$ and \mathbf{w}_0 is a fix previously trained model. Here, the separating hyperplane is described by the normal vector \mathbf{w} and the bias term b . The

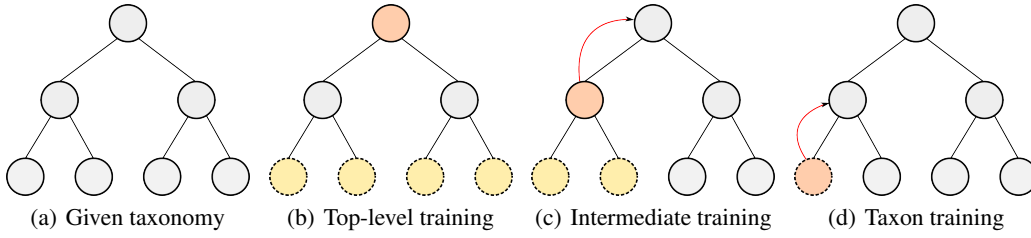


Figure 1: Illustration of the hierarchical top-down multitask training procedure. For this example, we consider four tasks and a given tree structure that relates them as shown in 1(a). Each leaf is associated with a training sample $S_i = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ that belongs to a particular task T_i . In 1(b), training begins by obtaining the predictor at the root node, for which data from all leaves is 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. data of all leaves below the current node. Additionally, 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 by only taking into account data for task T_1 to measure the loss, while again regularizing versus the parent predictor. Accordingly, the procedure is applied in a top-down manner to the remaining nodes until we have obtained a predictor for each leaf.

parameter C is used to trade off the loss on the training sample versus the regularization term to control generalization performance. In order to employ kernels, we derive the dual of the above formulation which is given by

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

$$\text{s.t. } 0 \leq \alpha_i \leq C \forall i \in \{1, n\}$$

$$\alpha^T \mathbf{y} = 0$$

where the α_i represent the dual variables of the current learning problem, whereas the α'_i are the dual variables obtained from the prior model $\mathbf{w}_0 = \sum_{i=1}^m \alpha'_i y'_i \mathbf{x}'_i$. In the standard SVM formulation, we have $p_i = -1 \forall i \in \{1, n\}$. In the 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* and *SVMLight* to handle prior information (will be made available later).

2.2 Top-Down Hierarchical Learning

We can now use the above framework to formulate an algorithm that directly exploits the structure of the given taxonomy. The idea is that the model at the parent $parent_{pred}(v)$ of each node v is used as prior information and is incorporated via the regularization term as described above. An illustration of the training procedure is given in Figure 1.

The training algorithm is formally described as Algorithm 1 below.

In a top-down manner a predictor f is obtained at each node v , where the loss L is evaluated on the union of training data $S = \cup_{\{S_i \preceq_{\mathcal{T}} v\}}$ at the leaves underneath the current node v , while the current model f is regularized versus the parent predictor f_p . Training is complete, once we have obtained a predictor f_i for each task T_i .

2.3 Multitask-Kernel Learning

We now turn to the two methods that depend on the previously discussed task similarity matrix γ , rather than the tree structure directly. First, we present *MultiKernel*, which is essentially a standard SVM with an extended kernel function:

Input: Training Samples S_i for $i \in \{1, M\}$, Taxonomy \mathcal{T}

Output: Trained Predictors $F = \{f_i : i \in \{1, M\}\}$

$F = \{\}$;

$V = \{v_{root}\}$;

while $|V| > 0$ **do**

$v = pop(V)$;

$S = \cup_{\{S_i \preccurlyeq_{\mathcal{T}} v\}}$;

$f_p = parentpred(v)$;

$f = \min_f R(f - f_p) + \sum_{(\mathbf{x}, y) \in S} L(y_i - f(\mathbf{x}_i))$;

if *isleaf*(v) **then**

$F = append(F, f)$

end

else

$V = append(V, children(v))$;

end

end

return F

Algorithm 1: Top-Down Training Procedure

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

where

$$\hat{k}(x_i, x_j) = \gamma_{task(x_i), task(x_j)} k(x_i, x_j).$$

Hence, we employ hierarchically related domains in order to design an extended kernel for task t that makes use of the data points of related tasks t' proportionally with respect to the similarity of tasks t and t' . The above formulation is an extension of the multitask approach presented in [2] and [3], which propose $\hat{k}(x_i, x_j) = (\frac{1}{\lambda} + \delta_{task(x_i), task(x_j)})k(x_i, x_j)$. The latter is a special case of our formulation, as we consider the general case of reweighing the original kernel k with a positive-semidefinite task-similarity matrix γ , which again yields a valid semi-definite kernel \hat{k} .

2.4 Pairwise Regularization

The following method *Pairwise* also depends on the task similarity matrix γ . The idea is that all classifiers $\mathbf{w}_1, \dots, \mathbf{w}_M$ are trained simultaneously, but the loss ℓ is measured separately for each \mathbf{w}_i on the corresponding sample S_i .

$$\min_{\mathbf{w}_1, \dots, \mathbf{w}_M} \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^M \gamma_{i,j} \|\mathbf{w}_i - \mathbf{w}_j\|^2 + C \sum_{i=1}^M \sum_{(\mathbf{x}, y) \in S_i} \ell(\langle \mathbf{x}, \mathbf{w}_i \rangle, y),$$

where ℓ is the hinge loss, $\ell(z, y) = \max\{1 - yz, 0\}$.

We cast this into a Multitask Learning formulation, by enforcing pairwise similarity between the \mathbf{w}_i . This is again achieved by using a modified regularization term. Again, the intuition is that tasks which are closely related are likely to have similar solutions. Thus, we use the task-similarity matrix γ to control how strongly we regularize each pair of $\mathbf{w}_i, \mathbf{w}_j$ to be close to each other.

Decomposition The above formulation cannot be easily solved with regular SVM implementations. Thus, we present 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 is met. It can be shown that the above optimization problem has a fixed point that coincides with the optimization problem of

$$\begin{aligned} \min_{\mathbf{w}_1, \dots, \mathbf{w}_M} \quad & \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^M \gamma_{i,j} \|\mathbf{w}_i - \mathbf{r}_i\|^2 + C \sum_{i=1}^M \sum_{(\mathbf{x}, y) \in S_i} \ell(\langle \mathbf{x}, \mathbf{w}_i \rangle, y), \\ \mathbf{r}_i = \quad & \sum_{j \neq i} \beta_{ij} \mathbf{w}_j, \\ \beta_{ij} = \quad & \frac{\gamma_{ij}}{\sum_{k \neq i} \gamma_{kj}}. \end{aligned}$$

This optimization states that each domain’s regularization prior \mathbf{r}_i should be in the convex hull of $\{\mathbf{w}_j\}_{j \neq i}$. This optimization problem can be solved using SVM implementations iteratively by optimizing each \mathbf{r}_i and \mathbf{w}_i until convergence. Investigating the relation between the dual and primal parameters,

$$\mathbf{w}_i = \sum_{t: x_t \in S_i} \alpha_t y_t x_t + \sum_{j \neq i} \gamma_{ij} \sum_{t: x_t \in S_j} \alpha_t y_t x_t,$$

reveals that, as previous methods, the pairwise regularization method yields *borrowing* support vectors of other domains with respect to the similarity measure.

3 Results

We performed two sets of experiments. The first set was performed on synthetic sequence data, which was created by applying mutations to a Position Specific Scoring Matrix (PSSM) according to a pre-defined tree structure (details omitted, code for data generation will be made available). Balanced, equally sized training data sets (100 examples) were sampled for each of the leaves. As test set, additional 5000 examples were sampled for each task. Next, we considered splice site data, similar to the data used in [6]. For each task, we obtained 10,000 training examples and an additional test set of 6,000 examples. In this dataset, there are 100 negative examples for each positive example, so we will report the area under the precision recall curve (auPRC), which is an appropriate measure for detection problems [5]. Experiments were performed for each of the presented Multitask methods and the following two baseline methods. In *Union*, all data are combined into one dataset $S = \cup_{i=1}^M S_i$ and a single global model is obtained that is used to predict on all domains. On the other extreme, we consider the baseline method *Plain*, where an individual SVM is trained on the data S_i of each domain separately, not taking into account data from the other domains. For each method, the cost constant C was optimized via cross-validation.

	<i>Plain</i>	<i>Union</i>	<i>MultiKernel</i>	<i>Pairwise</i>	<i>Hierarchical</i>
Toy data (auROC)	0.655	0.673	0.696	0.693	0.695
Splicing (auPRC)	0.469	0.505	0.545	0.532	0.563

Table 1: Tabular results, mean performance over tasks

On the toy dataset all three Multitask methods perform similarly well and perform clearly better than the two baselines on all eight tasks.

For the splice site dataset, we also observe superior performance of the Multitask methods. Here, the hierarchical method performs slightly better than the other Multitask methods. This is quite possibly due to a suboptimal choice of the similarity matrix (tree-hop-distance was used here).

In summary, we observe a clear improvement of the presented Multitask methods over the two baseline methods, for both datasets.

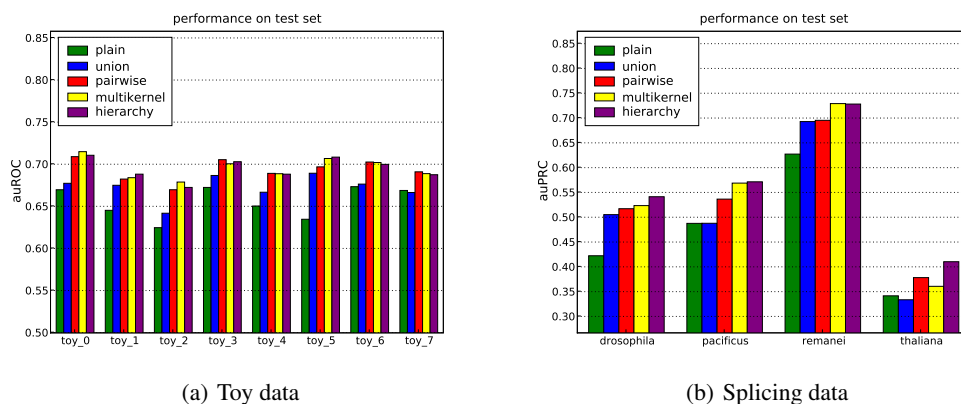


Figure 2: Results for individual tasks

4 Conclusion

We outlined two principle ways of leveraging a given taxonomy and presented three algorithms that readily deal with large scale problems such as those frequently encountered in genomic sequence analysis. We have demonstrated that our methods easily outperform baseline methods on synthetic data, and data from splice site prediction. On the one hand, the poor performance of *Plain* relative to the Multitask 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 trade off tasks according to their relatedness perform best.

We are encouraged by the good performance of the *Hierarchical* method, as it provides a fast, simple and non-parametric way of exploiting hierarchical information. Inferring an accurate task-similarity matrix γ proves to be non-trivial, therefore one should think of additional ways of using the hierarchy to ease that task. This is particularly the case, when edge lengths are unequal as can be observed from the splicing experiment. To cope with unequal edge lengths, the *Hierarchical* method could be extended to locally estimate the optimal degree of regularization towards the parent model by performing a cross-validation at each node v to determine an optimal C_v (which trades off loss versus regularization). That way, we use the hierarchy to restrict the hyper-parameter search space, as we do not have to search a grid of all combinations of parameters, but restrict ourselves to a linear search space per training step.

For all of the presented methods, we plan to provide publicly available scalable implementations based on modified versions of *SVMLight* and *LibSVM*. Lastly, we would like to emphasize that in Computational Biology there are a great number of problems that could be cast into a hierarchical Multitask learning scenario. Therefore, the methods and implementations that we presented can be employed for a wide range of problems.

References

- [1] 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.
- [2] H. Daumé. Frustratingly easy domain adaptation. In *ACL*. The Association for Computer Linguistics, 2007.
- [3] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.
- [4] 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.
- [5] G. Rätsch and S. Sonnenburg. *Accurate Splice Site Detection for Caenorhabditis elegans*. MIT Press, 2004.
- [6] 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, 2008.