JMLR: Workshop and Conference Proceedings 27:207–216, 2012 Workshop on Unsupervised and Transfer Learning

Multitask Learning in Computational Biology

Christian Widmer CWIDMER@CBIO.MSKCC.ORG and **Gunnar Rätsch** RAETSCH@CBIO.MSKCC.ORG
Friedrich Miescher Laboratory, Max Planck Society, Spemannstr. 39, 72076 Tübingen, Germany
Computational Biology Program, Sloan Kettering Institute, 1275 York ave, Box 357, New York City, NY 10065

Editor: I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver

Abstract

Computational Biology provides a wide range of applications for Multitask Learning (MTL) methods. As the generation of labels often is very costly in the biomedical domain, combining data from different related problems or tasks is a promising strategy to reduce label cost. In this paper, we present two problems from sequence biology, where MTL was successfully applied. For this, we use regularization-based MTL methods, with a special focus on the case of a hierarchical relationship between tasks. Furthermore, we propose strategies to refine the measure of task relatedness, which is of central importance in MTL and finally give some practical guidelines, when MTL strategies are likely to pay off.

Keywords: Transfer Learning, Multitask Learning, Domain Adaptation, Computational Biology, Bioinformatics, Sequences, Support Vector Machines, Kernel Methods

1. Introduction

In Computational Biology, supervised learning methods are often used to model biological mechanisms in order to describe and ultimately understand them. These models have to be rich enough to capture the often considerable complexity of these mechanisms. Having a complex model in turn requires a reasonably sized training set, which is often not available for many applications. Especially in the biomedical domain, obtaining additional labeled training examples can be very costly in terms of time and money. Thus, it may often pay off to combine information from several related tasks as a way of obtaining more accurate models and reducing label cost.

In this paper, we will present several problems where MTL was successfully applied. We will take a closer look at a common special case, where different tasks correspond to different organisms. Examples of such a setting are splice-site recognition and promoter prediction for different organisms. Because these basic mechanisms tend to be relatively well conserved throughout evolution, we can benefit from combining data from several species. This special case is interesting because we are given a tree structure that relates the tasks at hand. For most parts of the *tree of life*, we have a good understanding of how closely related two organisms are and can use that information in the context of Multitask Learning. To illustrate that the relevance of MTL to Computational Biology goes beyond the *organisms-as-tasks* scenario, we present a problem from a setting, where different tasks correspond to different related protein variants, namely Major Histocompatibility Complex (MHC)-I binding prediction.

Clearly, there exists an abundance of problems in Bioformatics that could be approached from the MTL point of view (e.g. Qi et al. (2010)). For instance, different tasks could correspond to different cell lines, pathways (Park et al., 2010), tissues, genes (Mordelet and Vert, 2011), proteins (Heckerman et al., 2007), technology platforms such as microarrays, experimental conditions, individuals, tumor subtypes, just to name a few. This illustrates that MTL methods are of interest to many applications in Computational Biology.

2. Methods

Our methods strongly rely on regularization based supervised learning methods, such as the Support Vector Machine (SVM) (Boser et al., 1992) or Logistic Regression. In its most general form, it consists of a loss-term that captures the error with respect to the training data and a regularizer that penalizes model complexity

$$J(\Theta) = L(\Theta|X,Y) + R(\Theta).$$

This formulation can easily be generalized to the MTL setting, where we are interested in obtaining several models parameterized by $\Theta_1, ..., \Theta_M$, where M is the number of tasks. The above formulation can be extended by introducing an additional regularization term R_{MTL} that penalizes the discrepancy between the individual models (Evgeniou et al., 2005; Agarwal et al., 2010).

$$J(\Theta_1, ..., \Theta_M) = \sum_{i=1}^{M} L(\Theta_i|X,Y) + \sum_{i=1}^{M} R(\Theta_i) + R_{MTL}(\Theta_1, ..., \Theta_M).$$

2.1. Taxonomy-based transfer learning: Top-down

In this section, we present a regularization-based method that is applicable when tasks are related by a taxonomy (Widmer et al., 2010a). Hierarchical relations between tasks are particularly relevant to Computational Biology where different tasks may correspond to different organisms. In this context, we expect that the longer the common evolutionary history between two organisms, the more beneficial it is to share information between these organisms. In the given taxonomy, tasks correspond to leaves (i.e. taxons) and are related by its inner nodes. The basic idea of the training procedure is to mimic biological evolution by obtaining a more specialized classifier with each step from root to leaf. This specialization is achieved by minimizing the training error with respect to training examples from the current subtree (i.e. the tasks below the current node), while similarity to parent classifier is enforced through regularization.

The primal of the Top-down formulation is given by

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

where ℓ is the hinge loss $\ell(z,y) = \max\{1-yz, 0\}$, $\mathbf{w}_p$ is the parameter vector from the parent model and $B \in [0,1]$ is the hyper-parameter that controls the degree of MTL regularization. The dual of the above formulation is given by (Widmer et al., 2010a)):

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

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

where n and m are the number of training examples at the current level and the level of the parent model, respectively.

The α_i represent the dual variables of the current learning problem, whereas the α'_j represent the dual variables obtained from the parent model $\mathbf{w}_p$; in the case of the linear kernel, it is described as $\mathbf{w}_p = \sum_{j=1}^{m}\alpha'_j y'_j \boldsymbol{x}'_j$. The resulting predictor is given by

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

To clarify the algorithm, an illustration of the training procedure is given in Figure 1. If the nodes in the taxonomy represent species, then interior nodes correspond to extant species. Thus, interior nodes are trained on the union of training data from descendant leaves (see Figure 1). In the dual of the standard SVM, the term $\mathbf{p}$ (corresponding to the p_i in the equation above) is set to $\mathbf{p} = (-1,\ldots,-1)^T$ (i.e. there is no parent model). In our extended formulation, the p_i is pre-computed and passed to the SVM-solver as the linear term of the corresponding quadratic program (QP). Therefore, the solution of the above optimization problem can be efficiently computed using well established SVM solvers. We have extended LibSVM, SVMLight and LibLinear to handle a custom linear term p and provide the implementations as part of the SHOGUN-toolbox (Sonnenburg et al., 2010).

2.2. Graph-based transfer learning

Following Evgeniou et al. (2005) we use the following formulation

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_M} \frac{1}{2}\sum_{s=1}^{M}\sum_{t=1}^{M} A_{st}\,||\mathbf{w}_s - \mathbf{w}_t||^2 + C\sum_{s=1}^{M}\sum_{(\boldsymbol{x},y)\in S}\ell\left(\langle\boldsymbol{x},\mathbf{w}_s\rangle + b, y\right), \tag{2}$$

where ℓ is the hinge loss $\ell(z,y) = \max\{1-yz,0\}$ and A_{st} captures the externally provided task similarity between task s and task t. Intuitively speaking, we want to penalize differences between parameter vectors more strongly, if we know a-priori that tasks are closely related. Following Evgeniou et al. (2005), we rewrite the above formulation using the graph Laplacian L

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_M} \sum_{s=1}^{M}\sum_{t=1}^{M} L_{st}\mathbf{w}_s^T\mathbf{w}_t + C\sum_{s=1}^{M}\sum_{(\boldsymbol{x},y)\in S}\ell\left(\langle\boldsymbol{x},\mathbf{w}_s\rangle + b, y\right), \tag{3}$$

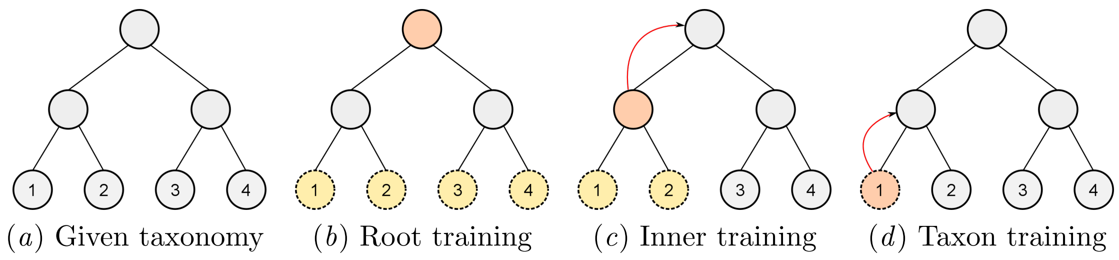


Figure 1: Illustration of the Top-down training procedure. In this example, we consider four tasks related via the tree structure in 1(a). Each leaf is associated with a task t and its training sample $S_t = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$. In 1(b), training begins by obtaining the predictor at the root node. Next, we move down one level and train a classifier at an inner node, as shown in 1(c). Here, the training error is computed with respect to $S_1 \cup S_2$, while the classifier *pulled* towards to the parent model $\mathbf{w}_p$ using the newly introduced regularization term, as indicated by the red arrow. Lastly, in 1(d), we obtain the final classifier of task 1 by only taking into account S_1 to compute the loss, while again regularizing against the parent predictor. The procedure is applied in a top-down manner to the remaining nodes until we have obtained a predictor for each leaf.

where $L = D - A$, where $D_{s,t} = \delta_{s,t} \sum_k A_{s,k}$. Finally, it can be shown that this gives rise to the following *multitask* kernel to be used in the corresponding dual (Evgeniou et al., 2005):

$$K((x, s), (z, t)) = L^+_{st} \cdot K_B(x, z), \tag{4}$$

where K_B is a kernel defined on examples and L^+ is the pseudo-inverse of the graph Laplacian. A closely related formulation was successfully used in the context of Computational Biology by Jacob and Vert (2008), where a kernel on tasks K_T was used instead of L^+, giving rise to

$$K((x, s), (z, t)) = K_T(s, t) \cdot K_B(x, z). \tag{5}$$

The corresponding joint feature space between task t and feature vector x can be written as a tensor product $\phi(t, x) = \phi_T(t) \cdot \phi_B(x)$ (Jacob and Vert, 2008). Note that a special case of this was presented by Daumé (2007) in the context of Domain Adaptation, where $\phi_T(t) = (1, 1, 0)$ was used as the source task descriptor and $\phi_T(t) = (1, 0, 1)$ for the target task, corresponding to $K_T(s, t) = (1 + \delta_{s,t})$.

2.3. Learning Task Similarity

An essential component of MTL is finding measure of task similarity that represents the improvement we expect from the information transfer between two tasks. Often, we are provided with external information on task relatedness (e.g. evolutionary history of organisms). However, the given similarity often only roughly corresponds to the task similarity relevant for the MTL algorithm and therefore one may need to find a suitable transformation to convert one into the other. For this, we propose several approaches, starting with a very simple parameterization of task similarities up to a sophisticated Multiple Kernel Learning (MKL) based formulation that learns task similarity along with the classifiers.

2.3.1. A simple approach: Cross-validation

The most general approach to refine a given task-similarity matrix $A_{s,t}$ is to introduce a mapping function ψ, parameterized by a vector Θ. The goal is to choose Θ such that the empirical loss is minimized. For example, we could choose a linear transformation $\psi_\Theta(s,t) = \Theta_1 + A_{s,t}\Theta_2$ or non-linear transformation $\psi_\Theta(s,t) = \Theta_1 \cdot exp(\frac{A_{s,t}}{\Theta_2})$. A straightforward way of finding a good mapping is using cross-validation to choose an optimal value for Θ.

A cross-validation based strategy may also be used in the context of the Top-down method. In principle, each edge e in the taxonomy could carry its own weight B_e. Because tuning a grid of all B_e in cross-validation would be quite costly, we propose to perform a *local* cross-validation and optimize the current B_e at each step from top to bottom independently. This can be interpreted as using the taxonomy to reduce the parameter search space.

2.3.2. Multitask Multiple Kernel Learning

In the following section, we present a more sophisticated technique to learn task similarity based on MKL. To be able to use MKL for Multitask Learning, we need to reformulate the Multitask Learning problem as a weighted linear combination of kernels (Widmer et al., 2010c). In contrast to Equation 5, the basic idea of our decomposition is to define task-based block masks on the kernel matrix of K_B. Given a list of tasks $\mathcal{T} = \{t_1, ..., t_T\}$, we define a kernel K_S on a subset of tasks $S \subseteq \mathcal{T}$ as follows:

$$K_S(x,y) = \begin{cases} K_B(x,y), & \text{if } t(x) \in S \wedge t(y) \in S \\ 0, & \text{else} \end{cases}$$

where $t(x)$ denotes the task of example x. Here, each S corresponds to a subset of tasks. In the most general formulation, we define a collection $\mathcal{I} = \{S_1, ..., S_p\}$ of an arbitrary number p of task sets S_i, which defines the latent structure over tasks. This collection leads to the following linear combination of kernels, which is positive semi-definite for $\beta_i \geq 0$:

$$\hat{K}(x,z) = \sum_{S_i \in \mathcal{I}} \beta_i K_{S_i}(x,z).$$

Using $\hat{K}$, we can readily utilize existing MKL methods to learn the weights β_i. This corresponds to identifying the groups of tasks S_i for which sharing information leads to improved performance. We use the following formulation of q-norm MKL by Kloft et al. (2011):

$$\begin{aligned} \min_{\boldsymbol{\beta}} \max_{\boldsymbol{\alpha}} \quad & \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \sum_{t=1}^{|\mathcal{I}|}\beta_t K_{S_t}(x_i,x_j) \\ \text{s.t.} \quad & ||\boldsymbol{\beta}||_q^q \leq 1, \boldsymbol{\beta} \geq 0 \\ & \mathbf{Y}^T\boldsymbol{\alpha} = 0, 0 \leq \boldsymbol{\alpha} \leq C \end{aligned}$$

To solve the above optimization problem, we follow ideas presented by Kloft et al. (2011) to iteratively solve a convex optimization problem involving only the β's and then to solve for α only. This method is known to converge fast even for a relatively large number of kernels (Kloft et al., 2011).

After training using MKL, a classifier f_t for each task t is given by:

$$f_t(z) = \sum_{i=0}^{N} \alpha_i y_i \sum_{S_j \in \mathcal{I}; t \in S_j} \beta_j K_{S_j}(x_i, z),$$

where N is the total number of training examples of all tasks combined. What remains to be discussed is how to define the collection $\mathcal{I}$ of candidate subsets S_i, which are subsequently to be weighted by MKL.

Powerset MT-MKL With no prior information given, a natural choice is to take into account all possible subsets of tasks. Given a set of tasks $\mathcal{T}$, this corresponds to the power set $\mathcal{P}$ of $\mathcal{T}$ (excluding the empty set) $\mathcal{I}_{\mathcal{P}} = \{S | S \in \mathcal{P}(\mathcal{T}) \wedge S \neq \emptyset\}$. Clearly, this gives us an exponential number (i.e. 2^T) of task sets S_i of which only a few will be relevant. To identify the relevant task sets, we propose to use an L1-regularized MKL approach to obtain a sparse solution. Most subset weights will be set to zero, leaving us with only a few relevant subsets with non-zero weights. By summing the weights of the groups containing a particular pair of tasks (s, t), we can recover the similarity $A_{s,t}$. As we do not assume prior information on task structure, this approach may be used to learn the task similarity matrix *de novo* (only for a small number of tasks due to computational complexity of the naïve implementation).

Hierarchical MT-MKL Next, we assume that we are given a tree structure $\mathcal{G}$ that relates our tasks at hand (see Figure 2). In this context, a task t_i corresponds to a leaf in $\mathcal{G}$. We can exploit the hierarchical structure $\mathcal{G}$ to determine which subsets potentially play a role for Multitask Learning. In other words, we use the hierarchy to restrict the space of possible task sets. Let $leaves(n) = \{l | l \text{ is descendant of } n\}$ be the set of leaves below the sub-tree rooted at node n. Then, we can give the following definition for the hierarchically decomposed kernel function

$$\hat{K}(x, y) = \sum_{n_i \in \mathcal{G}} \beta_i K_{leaves(n_i)}.$$

As an example, consider the kernel defined by a hierarchical decomposition according to Figure 2. In contrast to Power-set MT-MKL, we seek a non-sparse weighting of the task sets defined by the hierarchy and will therefore use $L2$-norm MKL (Kloft et al., 2011).

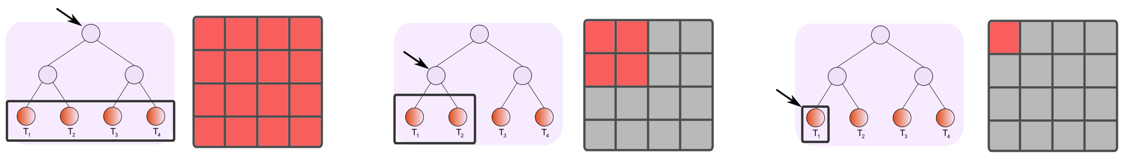


Figure 2: Example of a hierarchical decomposition. According to a simple binary tree, it is shown that each node defines a subset of tasks (a block in the corresponding kernel matrix on the left). Here, the decomposition is shown for only one path: The subset defined by the root node contains all tasks, the subset defined by the left inner node contains t_1 and t_2 and the subset defined by the leftmost leaf only contains t_1. Accordingly, each of the remaining nodes defines a subset S_i that contains all descendant tasks.

Alternatively, one can use a variation of the approach provided above and use boosting algorithms instead of MKL, as similarly done by Gehler and Nowozin (2009) and Widmer et al. (2010b).

2.4. When does MTL pay off?

In this section, we give some practical guide lines about when it is promising to use MTL algorithms. First, the tasks should neither be too similar nor too different (Widmer et al., 2010a). If the tasks are too different one will not be able to transfer information, or even end up with *negative transfer* (Pan and Yang, 2009). On the other hand, if tasks are almost identical, it might suffice to pool the training examples and train a single combined classifier. Another integral factor that needs to be considered is whether the problem is *easy* or *hard* with respect to the available training data. In this context, the problem can be considered *easy* if the performance of a classifier saturates as a function of the available training data. In that case using more out-of-domain information will not improve classification performance, assuming that none of the out-of-domain datasets are a better match for the in-domain test set than the in-domain training set.

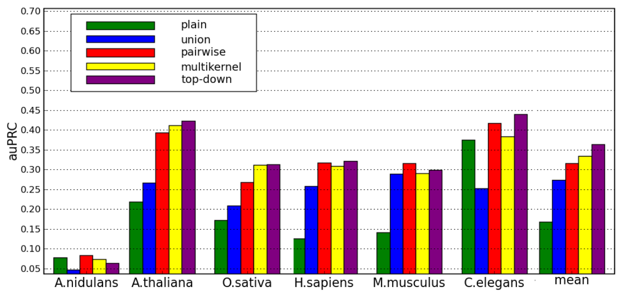
In order to control problem difficulty in the sense defined above, we can compute a learning curve (i.e. number of examples vs. auROC) and check whether we observe saturation. If this is the case, we do not expect transfer learning to be beneficial as model performance is most likely limited only by label noise. The same idea can be applied to empirically checking similarity between two tasks. For this, we compute pair-wise saturation curves (i.e. train on data from one task, test on the other) between tasks, giving us a useful measure to check whether transferring information between two tasks may be beneficial.

3. Applications

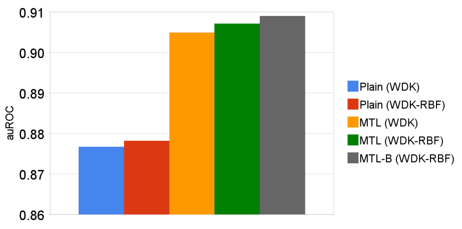
In this section, we give a two examples of applications in Computational Biology, where we have successfully employed Multitask Learning.

3.1. Splice-site prediction

The recognition of splice sites is an important problem in genome biology. By now it is fairly well understood and there exist experimentally confirmed labels for a broad range of organisms. In previous work, we have investigated how well information can be transfered between source and target organisms in different evolutionary distances (i.e. one-to-many) and training set sizes (Schweikert et al., 2009). We identified TL algorithms that are particularly well suited for this task. In a follow-up project we investigated how our results generalize to the MTL scenario (i.e. many-to-many) and showed that exploiting prior information about task similarity provided by taxonomy can be very valuable (Widmer et al., 2010a). An example how MTL can improve performance compared to baseline methods *plain* (i.e. learn a classifier for each task independently) and *union* (i.e. pool examples from all tasks and obtain a global classifier) is given in Figure 3(*a*). For an elaborate discussion of our experiments with splice-site prediction, please consider the original publications (Schweikert et al., 2009; Widmer et al., 2010a).



(*a*) Splicing Results

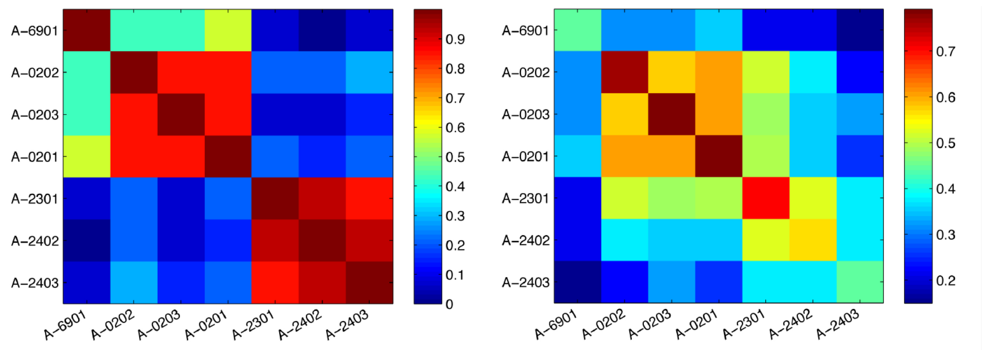


(*b*) MHC Results

Figure 3: In 3(*a*), we show results for 6 out of 15 organisms for baseline methods *plain* and *union* and three multitask learning algorithms. The mean performance is shown in the last column. 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). In 3(*b*), we report the average performance over the MHC-I alleles (tasks) as auROC for two baseline methods and three variants of MTL algorithms. Here, auROC was used to keep the results comparable to previously published results (Jacob and Vert, 2008).

3.2. MHC-I binding prediction

The second application we want to mention is MHC-I binding prediction. MHC class I molecules are key players in the human immune system. They bind small peptides derived from intracellular proteins and present them on the cell surface for surveillance by the immune system. Prediction of such MHC class I binding peptides is a vital step in the design of peptide-based vaccines and therefore one of the major problems in computational immunology. Thousands of different types of MHC class I molecules exist, each displaying a distinct binding specificity. We consider these different MHC types to be different tasks in a MTL setting. Clearly, we expect information sharing between tasks to be fruitful, if the binding pockets of the molecules exhibit similar properties. These properties are encoded in the protein sequence of the MHC proteins. Thus, we can use the externally provided similarity between MHC proteins to estimate task relatedness. In agreement with previous reports (Jacob and Vert, 2008), we observed that using the approach provided in Equation 5 with K_T defined on the externally provided sequences considerably boosts prediction accuracy compared to baseline methods, as can be seen in Figure 3(*b*). Lastly, we present an experiment, where we investigated how well a *learned* task-similarity (using Powerset MT-MKL) matches an externally provided one (i.e. based on sequence similarity). This externally provided task similarity is based on the Hamming-distance between the amino acids contained in the binding pockets of the MHC-I molecules. As can be seen in Figure 4, there is considerable agreement between the two similarity measures (i.e. an overall Spearman correlation coefficient of 0.67). From this, we conclude that MT-MKL may be of value in MTL settings, where relevant external information on the relatedness of tasks is absent.



(a) Sequence similarity between alleles (b) Task similarity computed from weights

Figure 4: Comparison of the task structure as identified by Powerset MKL with an externally provided one.

4. Conclusion

We have presented an overview of applications for MTL methods in the field of Computational Biology. Especially in the context of biomedical data, where generating training labels can be very expensive, MTL can be viewed as a means of obtaining more cost-effective predictors. We have presented several regularization-based MTL methods centered around the SVM, amongst them an approach for the special case of hierarchical task structure. Furthermore, we have outlined several strategies how to learn or refine the degree of task relatedness, which is of central importance when applying MTL methods. Lastly, we gave some practical guidelines to quickly check for a given dataset if one can expect MTL strategies to boost performance over straight-forward baseline methods.

Acknowledgments

We would like to acknowledge Jose Leiva, Yasemin Altun, Nora Toussaint, Gabriele Schweikert, Sören Sonnenburg, Klaus-Robert Müller, Bernhard Schölkopf, Oliver Stegle, Alexander Zien and Jonas Behr. This work was supported by the Max Planck Society, the German Research Foundation (grant RA1894/1-1) and the Sloan-Kettering Institute.

References

A. Agarwal, H. Daumé III, and S. Gerber. Learning Multiple Tasks using Manifold Regularization. In *NIPS Proceedings*. NIPS, 2010.

B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

H. Daumé. Frustratingly easy domain adaptation. In *Annual meeting-association for computational linguistics*, volume 45:1, page 256, 2007.

T. Evgeniou, C.A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6(1):615–637, 2005. ISSN 1532-4435.

P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Proc. ICCV*, volume 1:2, page 6, 2009.

D. Heckerman, C. Kadie, and J. Listgarten. Leveraging information across HLA alleles/supertypes improves epitope prediction. *Journal of Computational Biology*, 14(6): 736–746, 2007. ISSN 1066-5277.

L. Jacob and J. Vert. Efficient peptide-MHC-I binding prediction for alleles with few known binders. *Bioinformatics (Oxford, England)*, 24(3):358–66, February 2008.

M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien. lp-Norm Multiple Kernel Learning. *Journal of Machine Learning Research*, 12:953–997, 2011.

F. Mordelet and J.P. Vert. ProDiGe: PRioritization Of Disease Genes with multitask machine learning from positive and unlabeled examples. *Arxiv preprint*, 2011.

S.J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1345–1359, 2009.

C.Y. Park, D.C. Hess, C. Huttenhower, and O.G. Troyanskaya. Simultaneous genome-wide inference of physical, genetic, regulatory, and functional pathway components. *PLoS computational biology*, 6(11):e1001009, 2010.

Y. Qi, O. Tastan, J.G. Carbonell, J. Klein-Seetharaman, and J. Weston. Semi-supervised multi-task learning for predicting interactions between hiv-1 and human proteins. *Bioinformatics*, 26(18):i645, 2010.

G. Schweikert, C. Widmer, B. Schölkopf, and G. Rätsch. An Empirical Analysis of Domain Adaptation Algorithms for Genomic Sequence Analysis. In D Koller, D Schuurmans, Y Bengio, and L Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1433–1440. NIPS, 2009.

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. *Journal of Machine Learning Research*, 2010.

C. Widmer, J. Leiva, Y. Altun, and G. Rätsch. Leveraging Sequence Classification by Taxonomy-based Multitask Learning. In B. Berger, editor, *Research in Computational Molecular Biology*, pages 522–534. Springer, 2010a.

C. Widmer, N.C. Toussaint, Y. Altun, O. Kohlbacher, and G. Rätsch. Novel machine learning methods for MHC Class I binding prediction. In *Pattern Recognition in Bioinformatics*, pages 98–109. Springer, 2010b.

C. Widmer, N.C. Toussaint, Y. Altun, and G. Rätsch. Inferring latent task structure for Multitask Learning by Multiple Kernel Learning. *BMC bioinformatics*, 11 Suppl 8(Suppl 8):S5, January 2010c. doi: 10.1186/1471-2105-11-S8-S5.