

Generalizing to a zero-data task: a computational chemistry case study

Dumitru Erhan, Yoshua Bengio, Pierre-Jean L'Heureux
Dept. IRO, Université de Montréal
P.O. Box 6128, Downtown Branch, Montreal, H3C 3J7, QC, Canada
{erhandum,bengioy,lheureux}@iro.umontreal.ca

Shi Yi Yue
AstraZeneca R&D Montréal
7171 Frédérick Banting
St-Laurent, Québec, Canada, H4S 1Z9
ShiYi.Yue@astrazeneca.com

Technical Report 1286

Département d'Informatique et Recherche Opérationnelle

August 29th, 2006

Abstract

We investigate the problem of learning several tasks simultaneously in order to transfer the acquired knowledge to a completely *new task for which no training data are available*. Assuming that the tasks share some representation that we can discover efficiently, such a scenario should lead to a better model of the new task, as compared to the model that is learned by only using the knowledge of the new task. We have evaluated several supervised learning algorithms in order to discover shared representations among the tasks defined in a computational chemistry/drug discovery problem. We have cast the problem from a statistical learning point of view and set up the general hypotheses that have to be tested in order to validate the multi-task learning approach. We have then evaluated the performance of the learning algorithms and showed that it is indeed possible to learn a shared representation of the tasks that allows to generalize to a new task for which no training data are available. From a theoretical point of view, our contribution also comprises a modification to the Support Vector Machine algorithm, which can produce state-of-the-art results using multi-task learning concepts at its core. From a practical point of view, our contribution is that this algorithm can be readily used by pharmaceutical companies for virtual screening campaigns.

1 Introduction

Multi-task learning (MTL) is known under a variety of names: learning to learn, inductive transfer, bias learning, collaborative filtering etc. Each of these notions are variations of the same idea—that one can

construct learning techniques that can exploit acquired “knowledge” in order to bias the learning of a new task and thus improve the generalization performance.

It has already been shown theoretically and practically [1, 2] that taking into account multiple related tasks can be greatly beneficial to generalization, if the tasks are sufficiently related. If the added tasks are unrelated, the generalization power will usually decrease, but there are cases when even unrelated tasks might be helpful [3].

One popular application of MTL is Collaborative Filtering [4], which has its roots in recommender systems applications. Such systems produce recommendations that are based on similarities between the preferences of different users of the system. It is quite easy to draw the parallel between CF and MTL—predicting the preferences of one user is a single learning task, whereas modeling jointly the preferences of all the users of the system could be considered multi-task learning.

1.1 Motivation

One of the first stages of drug discovery is called High-Throughput Screening (HTS), during which a library of usually tens of thousands of compounds is tested against the target protein—which represents the “disease” that one tries to find a drug for—so that one can see how much the compounds influence this target. The process of physically testing each compound is time-consuming and expensive. Statistically reliable and computationally feasible methods for performing “virtual screens” of possibly computer-generated compounds, based on their molecular features or structures, are increasingly used in the pharmaceutical industry [5].

One interesting way of performing virtual screening more efficiently lies in exploiting the data from previous HTS campaigns. If these campaigns were performed on a set of related targets then it should be possible to transfer—in the inductive/multi-task learning sense—the knowledge acquired from the experiments to the virtual tests that are to be done on a new target (that is also related in some way to the targets that we have experiments for).

Interestingly, such multiple related targets do exist in the pharmaceutical industry, where they are commonly called a *target class*, e.g., kinases, G-protein coupled receptors, and maybe ion channels. These target classes have some common features. First, they represent some significant portion of a therapeutic area (in our case, the targets are related to the area of relieving “pain”). Some members of these target classes have been well studied. Second, targets within each of these target classes share a common structural frame. Members of each target class may have a similar binding site¹. Third, with the development of genomic projects, many new members of these target classes have been identified, though the biological roles of these newcomers (so called orphans) are still unknown. The challenge we are facing here is how to transfer our knowledge from known targets to orphans in a statistically reliable way.

The traditional statistical approach considers a different machine learning task for each member of a given class. It is also proposed in the drug discovery literature that we gain confidence on a specific relationship when the same methodology produces concordant results on similar tasks [6]. We would like to extend that with the concepts from Multi-Task Learning.

The parallel between collaborative filtering/multi-task learning and virtual screening can be made almost immediately: the “users” (or the “tasks”) are the biological targets, the “items” (or the “inputs”) are the molecular compounds and the “ratings” (or the “outputs”) are the levels of activity of the given compound for a given target. The descriptors or the features of the targets and of the compounds could be anything that might help us in uncovering relationships both between the targets and the compounds.

¹The region on the target to which specific compounds form chemical bonds.

1.2 Contributions of this paper

This paper investigates several questions related to the process of multi-task learning. First of all, we were interested in developing practical methods for measuring the degree to which we can profit from learning multiple tasks at the same time. We are also interested in how it is possible to “transfer” the “knowledge” acquired by learning several tasks to a *completely new task*. It is also very interesting for us to see the evolution of the generalization of a multi-task learning algorithm when trained with only a small sample of data from a given task, as a function of the size of this sample. Finally, we wanted to explore theoretical and practical ways of encoding the similarity or the relatedness of tasks and exploiting this for improving the multi-task learning algorithms that we used.

To this end, we have defined a clear way of testing, for a particular dataset, the degree to which multi-task learning helps, when compared to standard single-task learning. To the best of our knowledge, this paper is the first to investigate thoroughly the problem of generalizing an MTL algorithm to a completely new task. We have demonstrated that such “pure” inductive transfer is possible and it can be quite helpful. All the algorithms that we used have built-in ways of computing a similarity measure between tasks; one of these algorithms, the Multi-Task Support Vector Machine that uses the Collaborative Filtering kernel matches the state-of-the-art performance and is a clear candidate for inclusion into the industrial process of drug discovery.

From the point of view of the drug discovery process, our objective and contribution was to compare and evaluate methods to take advantage of the commonalities between the different targets within a target class. In addition, we developed a solution that allows us to estimate Quantitative structure-activity relationship (QSAR)² models for orphans that have not yet been tested, or for which there are very little available data. We have developed a practical approach where very little prior knowledge of the target is needed; we were less interested in building the best model for a single target than building a model for which we lack sufficient data. Finally, to the best of our knowledge, such a *multi-target* dataset has never been discussed before in the computational chemistry literature. This paper is therefore the first to offer insights into this kind of a dataset and ways to solve problems defined by it.

2 Previous work

The first attempts at performing multi-task learning were by extending the standard multi-layer neural networks[7, 3]. The simplest of such extensions was to create a shared hidden layer that is trained in parallel for all the tasks. Such extensions performed inductive bias into the learning procedure and improved the generalization error. A parallel development[1] used Bayesian methods to construct a setting that would resemble sampling different learning tasks from the environment. The same paper gave bounds on the information needed to learn a task, when it is learned concurrently with other tasks. Baxter[8] expanded on that and gave bounds on the number of tasks that are sufficient in order to learn a novel task. Ben-David and Schuller[9] demonstrated a set of simple transformations that defined *task relatedness*. Evgeniou and Pontil [10] described ways to use kernel methods in order to perform multi-task learning and one of their ideas provides for task-related features that could help during the learning process.

Our work builds on the ideas from the above: we are interested in measuring task relatedness, in learning a completely new task, in using task-specific features to encode task relatedness and in devising techniques for improving the generalization performance while using multi-task learning for a specific

²The general notion of linking properties of molecular compounds to their behavior in the presence of certain targets

computational chemistry dataset. Our application of multi-task learning techniques to such a dataset has never been done, to the best of our knowledge. However, the techniques that we employed are simple extensions to those that are popular in the industry[5].

Quantitative structure-activity relationship emerged as a valuable technique for the pharmaceutical sciences some thirty years ago [11, 6]. It has since evolved into many branches of research (review in [12]) and surfed on the growing capabilities of computers, like neural networks [13] and 3D-QSAR [14]. In recent years, several other groups have introduced kernel machines [15] and Support Vector Machines in QSAR [16, 17]. It has often proved superior to Partial Least Square or neural networks on descriptor-based relationships. It has sparked renewed interest in similarity measures specific to the chemical settings [18].

3 Problem definition and proposed algorithms

Assume a collection of k datasets, where each dataset corresponds to a (classification, regression etc.) task that is to be solved. Each of these datasets consists n_k pairs $(x_i^k, y_i^k), i = 1 \dots n_k$, where $x_i^k \in \mathbb{R}^{D_x}$ and $y_i^k \in \mathbb{R}$ (the x_i^k can overlap across datasets). Assume that for each of these datasets we are also given a vector $t^k \in \mathbb{R}^{D_k}$, which is a set of task-specific descriptors or features. We are interested in finding algorithms that would be able to exploit this data in such a way such that when presented with a new dataset of n_{new} pairs (x_i^{new}, y_i^{new}) and a vector $t^{new} \in \mathbb{R}^{D_{new}}$, they would be able to generalize well to this dataset, i.e. predict y_i^{new} well, according to some loss function, given x_i^{new} and t^{new} . The algorithm must generalize well *without having seen* any of the $(x_i^{new}, y_i^{new}, t^{new})$ triplets or after seeing a very small sample of the triplets from the new task.

3.1 Multi-Task Neural Network

The basic architecture of the neural network model (Figure 1) has two hidden layers. The first hidden layer is committed to discovering a low-dimensional embedding of the tasks, i.e. to discover a shared representation (for the biological target descriptors, for instance).

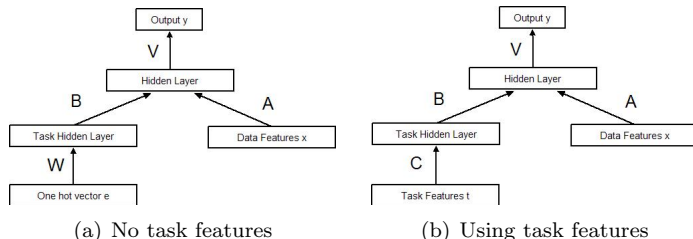


Figure 1: Multi-task Neural Network architectures

In one version we use in input of the first layer a one-hot variable (a vector e_i full of zeros except for a 1 at position i) indicating the task. The second layer receives the output of the first layer and the features of the given data point. This architecture will learn an individual predictive model for each task, but the first layer will contain information about the relatedness of task. Here is the equation that corresponds to the model depicted on Figure 1a:

$$P(y_i = 1|x_i) = \text{sigmoid}(V \tanh(A x_i + B \tanh(W e_i)))$$

In another version, depicted on Figure 1b, we use task descriptors t_i in the first layer. Here we learn an indirect predictive model for each task. Here’s the equation that corresponds to this model:

$$P(y_i = 1|x_i, t_i) = \text{sigmoid}(V \tanh(A x_i + B \tanh(C t_i)))$$

The learned parameter matrices are V, A, B, W and C .

3.2 A collaborative filtering kernel

The idea of having a shared representation of the tasks is very close to the one of using kernels in order to encode the similarity between tasks. We could combine the similarity measure between tasks with the one between the data points in order to create a similarity measure between task-datapoint pairs.

The idea of using such a kernel belongs to [19], who used it in a collaborative filtering setting. The algorithm that was used is the so-called Joint Perceptron Ranker (JRank). Reference [19] contains a more detailed description, so we present only the key ideas. For two given task-datapoint pairs (\mathbf{t}, \mathbf{x}) and $(\mathbf{t}', \mathbf{x}')$, we define the kernel as

$$K((\mathbf{t}, \mathbf{x}), (\mathbf{t}', \mathbf{x}')) = \Psi(\mathbf{t}, \mathbf{x})^T \cdot \Psi(\mathbf{t}', \mathbf{x}'). \quad (1)$$

where Ψ is the corresponding feature map. To define a kernel over task-datapoint pairs, we first define similarity measures between pairs of tasks, then between pairs of datapoints, and then combine the two measures into a kernel function of the desired type.

To combine the two measures into one, we could use the tensor product between the task kernel and the data kernel to get $K((\mathbf{t}, \mathbf{x}), (\mathbf{t}', \mathbf{x}'))$. However, it can be shown that this product is equivalent to

$$K((\mathbf{t}, \mathbf{x}), (\mathbf{t}', \mathbf{x}')) = K_{\mathcal{T}}(\mathbf{t}, \mathbf{t}') \cdot K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') \quad (2)$$

(where $K_{\mathcal{T}}$ and $K_{\mathcal{X}}$ are the kernels for tasks and datapoints, respectively; these kernels could be any standard kernels or valid combinations thereof) which is a handy shortcut.

Given this kernel it is now easy to define an algorithm that will use it. For more details regarding JRank see [19]. Our implementation follows closely the one described in the original paper, except that we do not solve an ordinal regression problem, but a simple binary classification problem.

3.3 Multi-Task Support Vector Machines

Once there is a kernel defined, we can use the standard SVM algorithm[20].

The algorithm has several hyper-parameters: the ones related to the underlying kernels and the soft-margin parameter C . Thus, once we have the triples (task,datapoint,output) and once we define the kernel we can use any standard implementation of an SVM algorithm to solve the optimization problem (we use SVMlight[21]).

4 Experimental setup

4.1 Dataset

AstraZeneca R&D Montreal amassed the raw data used for this study through internal and external HTS. The compound descriptors were computed with MOE [22]. The set of 469 descriptors per compound

range from atom frequencies and topological indices to 3D surface area descriptors. We also computed MACCS, Randic and EState descriptors that are available in the MOE package. The numerical values were normalized.

Several receptor-binding fingerprints (for the targets) have been published. The binding pocket fingerprints we used in this work were based on our observations and some assumptions. When a ligand interacts in the binding site, it will use some of the native interactions and also build up interactions specific for the ligand. Based on the positions (at the binding site of each target put in a common reference position), the type of interactions at this position (ionic, polar or hydrophobic), and the various amount of targets using those positions, we collected 14 bins. Each bin represents a type of the interaction at the given position of the binding pocket.

There are 24 targets and approx. 40,000 compounds. Not all the compounds have been tested with all the targets, resulting in approx. 186,000 tests. The number of compounds tested per target varies greatly. The proportion of active compounds (for a given target) is usually very small, so the algorithms have to account for that.

4.2 Task selection

Given the size of the dataset (186,000 tests) we were interested in finding ways to select several of the more "related" targets so that we can concentrate on them. In order to do that, we computed the pairwise linear correlation of activity between each target, for shared chemical compounds in the dataset.

Many of the pairs' correlations are small. From the 24 targets, we picked 7 targets, for which cross-correlation where the strongest. These seven targets—A, D, F, H, I, S and U—constituted our main focus of interest.

4.3 Measuring performance

Lift We use the so-called "lift" to assess the performance of the models. The lift measures how much better than random we can order the compounds (order them in decreasing order of probability to be an active compound) and is a better measure of performance than the binary classification error in case the dataset is very unbalanced (which is our case). Here a/n is the average fraction of actives obtained by taking a random subset of compounds, with a the total number of actives, n the total number of compounds. In the selected subset, a_s is the number of actives and n_s the number of compounds in that subset.

$$\text{LIFT} = 100 \cdot \frac{a_s/n_s}{a/n} \quad (3)$$

The lift that we compute corresponds to a single point on an enrichment curve (and is roughly proportional to a point on an ROC curve). The size of subset is 30%, which is a standard size in the computational chemistry literature. The higher the lift value, the better is the performance of the algorithm.

Undersampling In order to recreate an environment where we lack sufficient number of examples to learn a predictive model, we artificially deplete a dataset, focusing on a target. Focusing on each of the 7 targets at a time, we split randomly the dataset such that the training set contains none or a small fraction of the training data for that target, plus all the training data for the 6 other targets.

Hyper-parameter selection The parameters of the neural network (V, A, B, W) or (V, A, B, C) are tuned by stochastic gradient ascent on the average conditional log-likelihood (cross-entropy) on the training set. The number of hidden units (e.g., dimension of A) and optimization parameters are selected using a validation set disjoint from the test set.

In the case of JRank, the widths of the Gaussian kernels (we used one for the task features and data features, respectively) were computed by cross-validation, whereas early stopping was used for finding an optimal number of iterations. For the Multi-Task Support Vector Machines, we used cross-validation to pick the best σ s and soft-margin parameter C . As with JRank, we used two Gaussian kernels.

5 Experimental results

The first set of experiments was to test the performance of the learning algorithms when we train on 6 targets and test on a 7th, i.e. by having *zero* training examples from the 7th target. Table 1 shows a summary of the results. For 3 out of 7 targets, the Multi-Task Support Vector Machine generalized quite well, with lifts in the range of 130–161 (where 100 is the lift of a random decision algorithm). JRank and MT-NNet did not seem to be able to do the same, with lift values hovering around the 100 value.

The results obtained with MT-SVM are the key findings of this paper: they show pretty clearly that in the case of MT-SVM and of this dataset, it is *possible* to transfer some of the “knowledge” acquired from learning a set of tasks to a completely new one with zero training examples.

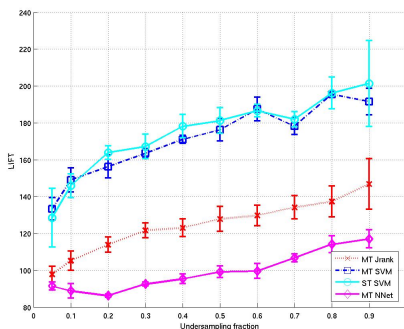
Target	MT-SVM	JRank	MT-NNet
A	105	102	102
D	108	99	95
F	150	108	101
H	161	110	105
I	130	104	103
S	106	105	102
U	105	105	98

Table 1: Lifts obtained by testing on a completely new target with no training data

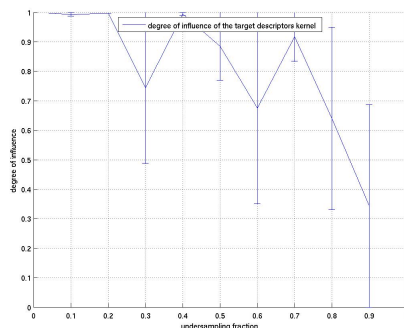
The next step that we undertook was to perform undersampling for each of the 7 targets and compare the performance of the algorithms presented above. Figure 2a compares the Multi-Task Neural Network, JRank in its multi-task version and Multi-Task Support Vector Machines. It also plots the performance of the simple Support Vector Machine algorithm that is trained with only the undersampling fraction of data from the given target and tested with the rest of it (the difference being that in the other cases the algorithms are trained with the data from the rest of the targets as well). We present the performance for one of the targets—in case of the rest of targets, the plots are very similar in nature.

It is clear that the only match to the simple SVM algorithm is MT-SVM. That is a good achievement, because the normal SVM is usually the state-of-the-art in computational chemistry/drug discovery applications. Unfortunately, MT-SVM is not able to out-perform the SVM at any undersampling fraction, not even at the smaller ones, contrary to our expectations.

Having target descriptors seems to help the performance in the case of the Neural Network and JRank.



(a) Comparison of the algorithms on target U



(b) Influence of target I descriptors

Figure 2: Experiments with undersampling

Both of the algorithms can build a model without these descriptors, but their presence is definitely making a difference.

We have also computed an estimate of the “degree of influence” of target descriptors. We were interested in finding out how much they influence the decision making process, as a function of the undersampling fraction. We had expected that this influence would decrease as we add more and more data from the given target and that it would be maximal when the undersampling fraction is small. This estimate can be computed by simply observing that, if the task kernel and the data kernel are both Gaussians, then the resulting task-datapoint kernel will be Gaussian as well. The exponent will be a sum that has a term for the task and a term for the datapoint. The relative fraction of the task-related term (when applying the kernel, for instance, at the test stage) will give us an idea of the level of influence of the task features. Figure 2b shows such a plot for target I. This plot (as well as the plots for 3 other targets) confirms our expectations stated above.

6 Future work

It seems that in order for MT-SVM to make better use of the other tasks one would have to attempt the following remedies:

- Try out different target descriptors. We have already seen that their mere presence helps with learning, therefore if we were to get higher quality descriptors, MT-SVM should perform better.
- Use data from all 24 targets for training. This could help, but it could also have negative effects, because of the fact that the targets might not necessarily be related (and the noise introduced in this way could be harmful).
- We have not tried other kernels, except the Gaussian one. It is quite possible that a polynomial kernel or a linear combination of several different kernel could help the process.

On a more general note, one could try extending the neural network architecture, by predicting not only the activity but also the target and compound features. If constructed properly, such a network could profit from the inductive bias of these features. One could also try to apply the ideas from [10] to this problem: they present an SVM model that could profit from task/target descriptors. Bakker and Heskes’s Bayesian way of multi-task learning[23] can accommodate for such descriptors, too, therefore it is an option to consider for further research. Finally, one should consider coming up with a generative model, instead of a purely discriminative one. Intuitively, it seems that a generative model could cope better with the concept of generalizing to a completely new task for which there are no training examples at all.

7 Conclusions

We have evaluated several machine learning algorithms that we used to solve a computational chemistry problem. We were interested in studying the behavior of these algorithms when presented with a completely new and unseen task (or target). Our goal was to design an algorithm that would be able to transfer the knowledge acquired from learning several (possibly related) task to a new task. To this end, we designed a Multi-Task Neural Network, tested a kernel-based method called JRank and applied the ideas from JRank to the standard Support Vector Machine algorithm (we call this extension the Multi-Task Support Vector Machine).

The Multi-Task Support Vector Machine algorithm matches the performance of the state-of-the-art algorithm for the given problem, at all undersampling fractions. It also manages to achieve inductive transfer when tested on several unseen tasks/targets, which is an encouraging result. It makes us believe that such inductive transfer is possible in general (not only with these targets). We have suggested possible avenues for improvement and traced parallels to other published algorithms that could make intelligent use of our dataset.

References

- [1] Jonathan Baxter. A bayesian/information theoretic model of bias learning. In *COLT '96: Proceedings of the ninth annual conference on Computational learning theory*, pages 77–88, New York, NY, USA, 1996. ACM Press.
- [2] J. Ghosh and Y. Bengio. Bias learning, knowledge sharing. *IEEE Transactions on Neural Networks*, 14(4):748–765, Jul 2003.

- [3] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.
- [4] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM Press.
- [5] H.J. Bhm and G. Schneider, editors. *Virtual Screening for Bioactive Molecules*, volume 10 of *Methods and Principles in Medicinal Chemistry*. WILEY-VCH, Weinheim, Germany, 2000.
- [6] C. Hansch and A. Leo. *Exploring QSAR: Fundamentals and Applications in Chemistry and Biology*. ACS Professional Reference Book, 1995.
- [7] Rich Caruana. Multitask learning: A knowledge-based source of inductive bias. In *International Conference on Machine Learning*, pages 41–48, 1993.
- [8] Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.
- [9] S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *COLT '03: Proceedings of the Sixteenth Annual Conference on Learning Theory*, 2003.
- [10] T. Evgeniou, C. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.
- [11] R. Smith, C. Hansch, and M. Ames. Selection of a reference partitioning system for drug design work. *Journal of Pharmaceutical Science*, 64:599–606, 1975.
- [12] H. Kubinyi. Qsar and 3d qsar in drug design part 1: methodology. *Drug Discovery Today*, 2:457–467, 1997.
- [13] J. Zupan and J. Gasteiger. *Neural Networks in Chemistry and Drug Design*. Wiley-VCH, Weinheim, Germany, 2nd edition, 1999.
- [14] H. Kubinyi, G. Folkers, and Y.C. Martin, editors. *3D QSAR in Drug Design*, volume 3. Kluwer Academic Publishers, 1998.
- [15] Michinari Momma and Kristin P. Bennett. Sparse kernel partial least squares regression. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *COLT*, volume 2777 of *Lecture Notes in Computer Science*, pages 216–230. Springer, 2003.
- [16] M. K. Warmuth, J. Liao, G. Ratsch, M. Mathieson, S. Putta, and C. Lemmen. Active learning with support vector machines in the drug discovery process. *Journal of Chemical Information and Modeling*, 43(2):667–673, 2003.
- [17] K. R. Muller, G. Ratsch, S. Sonnenburg, S. Mika, M. Grimm, and N. Heinrich. Classifying 'drug-likeness' with kernel-based learning methods. *Journal of Chemical Information and Modeling*, 45(2):249–253, 2005.
- [18] H. Frohlich, J.K. Wegner, F. Sieker, and A. Zell. Kernel functions for attributed molecular graphs - a new similarity-based approach to adme prediction in classification and regression. *Qsar & Combinatorial Science*, 25:317–326, 2006.
- [19] Justin Basilico and Thomas Hofmann. Unifying collaborative and content-based filtering. In Carla E. Brodley, editor, *ICML*. ACM, 2004.
- [20] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.
- [21] Thorsten Joachims. *Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed.). MIT Press, 1999.
- [22] Chemical Computing Group. MOE. www.chemcomp.com, 2004.
- [23] Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.