# Multilabel Classification of Drug-like Molecules via Max-margin Conditional Random Fields

Hongyu Su
University of Helsinki, Finland
hongyu.su@cs.helsinki.fi

Markus Heinonen
University of Helsinki, Finland
markus.heinonen@cs.helsinki.fi

Juho Rousu
University of Helsinki, Finland
juho.rousu@cs.helsinki.fi

**Abstract**

We present a multilabel learning approach for molecular classification, an important task in drug discovery. We use a conditional random field to model the dependencies between drug targets and discriminative training to separate correct multilabels from incorrect ones with a large margin. Efficient training of the model is ensured by conditional gradient optimization on the marginal dual polytope, using loopy belief propagation to find the steepest feasible ascent directions. In our experiments, the MMCRF method outperformed the support vector machine with state-of-the-art graph kernels on a dataset comprising of cancer inhibition potential of drug-like molecules against a large number cancer cell lines.

## 1 Introduction

Machine learning has become increasingly important in drug discovery, where viable molecular structures are searched or designed for therapeutic efficacy. In particular, the costly preclinical *in vitro* and *in vivo* testing of drug candidates can be focused to the most promising molecules, if accurate *in silico* models are available (Trotter et al., 2001).

Molecular classification has been tackled with a variety of methods, including inductive logic programming (King et al., 1996) and artificial neural networks (Bernazzani et al., 2006). During the last decade kernel methods (Ralaivola et al., 2005; Swamidass et al., 2005; Trotter et al., 2001; Ceroni et al., 2007) have emerged as a computationally effective way to handle the non-linear properties of chemicals. In numerous studies, SVM-based methods have obtained promising results (Byvatov et al., 2003; Trotter et al., 2001; Zernov et al., 2003). However, classification methods focusing on a single target variable are probably not optimally suited to drug screening applications where large number of target cell lines are to be handled.

Multilabel classification, where the objects can be classified into more than one category at a time, have received a significant attention in recent years both in hierarchical (Silla and Freitas, 2010) and Bayesian network settings (de Waal and van der Gaag, 2007; Rodriguez and Lozano, 2008). In this paper we propose, to our knowledge, the first application of multilabel learning to molecular classification. Our learning method belongs to the structured output prediction family (Taskar et al., 2003; Tsochantaridis et al., 2004; Rousu et al., 2006; Rousu et al., 2007); the drug targets (cancer cell lines) are organized in a Markov network, drug molecules are represented by kernels and max-margin training is used to learn the

parameters. Loopy belief propagation over the Markov network is used both in learning the model and in extracting the predicted multilabel.

## 2 Multilabel learning with MMCRF

The model used in this paper is an instantiation of the Max-Margin Conditional Random Field (MMCRF) framework (Rousu et al., 2007) for associative Markov networks and can also seen as a sibling method to HM$^3$ (Rousu et al., 2006), which is designed for hierarchies. Here we give an overview of the method for transparency, the interested reader may check the details from the above references.

We consider data from a domain $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is a set and $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k$ is a Cartesian product of the sets $\mathcal{Y}_j = \{+1,-1\}, j = 1, \ldots, k$. A vector $\mathbf{y} = (y_1, \ldots, y_k) \in \mathcal{Y}$ is called the *multilabel* and the components $y_j$ are called the *microlabels*.

We assume that a training set $\{(x_i, \mathbf{y}_i)\}_{i=1}^m \subset \mathcal{X} \times \mathcal{Y}$ has been given. In addition, a pair $(x_i, \mathbf{y})$ where $x_i$ is a training pattern and $\mathbf{y} \in \mathcal{Y}$ is arbitrary, is called a *pseudo-example*, to denote the fact that the pair may or may not be generated by the distribution generating the training examples. As the model class we use the exponential family

$$P(\mathbf{y}|x) \propto \prod_{e \in E} \exp\left(\mathbf{w}_e^T \varphi_e(x, \mathbf{y}_e)\right)$$

defined on the edges of a Markov network $G = (V, E)$, where node $j \in V$ corresponds to the $j$'th component of the multilabel and the edges $e = (j, j') \in E$ correspond to a microlabel dependency structure given as input. By $\mathbf{y}_e = (y_j, y_{j'})$ we denote the pair of microlabels of the edge $e = (j, j')$ and $\varphi(x, \mathbf{y}) = (\varphi_e(x, \mathbf{y}_e))_{e \in E}$ is a *joint feature map* for inputs and outputs. The joint feature map is given by the tensor product $\varphi(x, \mathbf{y}) = \phi(x) \otimes \psi(\mathbf{y})$ of input features $\phi(x)$ computed from the molecules (see Section 3) and output features $\psi(\mathbf{y}) = (\psi_{eu}(\mathbf{y}))$ corresponding to possible labelings $u \in \{+1, -1\}^2$ of the edges $e \in E$: $\psi_{eu}(\mathbf{y}) = [\![\mathbf{y}_e = u]\!]$. The tensor product then contains all pairs $\phi_r(x) \cdot \psi_{eu}(\mathbf{y})$ of input and output features. The benefit of the tensor product representation is that context (edge-labeling) sensitive weights can be learned for input features and no prior alignment of input and output features need to be assumed.

### 2.1 Max margin learning

To learn the parameters of the model we apply margin-based structured output prediction (c.f. (Taskar et al., 2003; Tsochantaridis et al., 2004)). The primal optimization problem takes the form

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \ & \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^m \xi_i \\ \text{s.t. } & \mathbf{w}^T \Delta\varphi(x_i, \mathbf{y}) \geq \ell(\mathbf{y}_i, \mathbf{y}) - \xi_i, \\ & \text{for all } i \text{ and } \mathbf{y}, \end{aligned} \tag{1}$$

where $\Delta\varphi(x_i, \mathbf{y}) = \varphi(x_i, \mathbf{y}_i) - \varphi(x_i, \mathbf{y})$, $\ell(\mathbf{y}_i, \mathbf{y})$ is the loss of the pseudo-example, $\xi_i$ is the slack and the parameter $C$ controls the amount of regularization in the model. The corresponding Lagrangian dual problem takes the form:

$$\begin{aligned} \underset{\alpha \geq 0}{\text{maximize}} \ & \alpha^T \ell - \frac{1}{2} \alpha^T K \alpha \\ \text{s.t. } & \sum_{\mathbf{y}} \alpha(i, \mathbf{y}) \leq C, \forall i, \mathbf{y}, \end{aligned} \tag{2}$$

where $K = \left(\Delta\varphi(x_i, \mathbf{y})^T \Delta\varphi(x_j, \mathbf{y}')\right)$ is the *joint* kernel matrix for *pseudo-examples* $(x_i, \mathbf{y})$ and $\ell = (\ell(\mathbf{y}_i, \mathbf{y}))_{i,\mathbf{y}}$ encodes the loss for each $(x_i, \mathbf{y})$.

### 2.2 Marginal dual problem

The above optimization problems are challenging due to the exponential number of constraints or dual variables. A more manageable sized problem is obtained by considering the edge-marginals of dual variables

$$\mu(i, e, u) = \sum_{\mathbf{y} \in \mathcal{Y}} \psi_{eu}(\mathbf{y}) \alpha(i, y), \tag{3}$$

where $e \in E$ is an edge in the output network and $u \in \{+1, -1\}^2$ is a possible labeling for the edge. Using the marginal dual representation, we can state the dual problem (2) in equivalent form as (for details, see (Rousu et al., 2007)):

$$\max_{\mu \in \mathcal{M}^m} \ \mu^T \ell - \frac{1}{2} \mu^T K_E \mu, \tag{4}$$

where $\mathcal{M}$ denotes the marginal polytope[1] (c.f. (Wainwright et al., 2005)), the set of all combinations of marginal variables (3) of an example that have a counterpart in the dual feasible set in (2), $K_E = diag(K_e)_{e \in E}$ contains the joint kernel values pertaining to the edges, $\mu = (\mu(i, e, v))$ is the vector of marginal dual variables and $\ell = (\ell(i, e, v))$ is the vector of losses between edge-labelings, $\ell(i, e, v) = \ell(\mathbf{y}_{ie}, v)$. This problem is a quadratic programme with a number of variables *linear* in both the size of the output network and the number of training examples. Thus, there is an exponential reduction in the number of dual variables from the original dual (2).

### 2.3 Conditional gradient optimization

The marginal dual problem is solved by an iterative optimization algorithm where the marginal dual variables of each example in turn are optimized using conditional gradient algorithm whilst keeping the other training examples fixed. The conditional gradient step (Algorithm 1) iteratively finds the best feasible direction given the current subgradient $g_i = \ell_i - K_{i\cdot}\mu$ of the objective

$$\mu_i^* = \underset{\mathbf{v} \in \mathcal{M}}{\mathbf{argmax}}\ g_i^T \mathbf{v} \tag{5}$$

and uses exact line search to locate the optimal point in that direction.

The feasible ascent directions in (5) correspond to vertices[2] of the marginal dual polytope $\mathcal{M}$ which via (3) are images of the vertices of the original dual set and thus have one to one correspondence to the set of possible multilabels.

Consequently, for each solution $\mu_i^*$ of (5) there is a corresponding multilabel $\mathbf{y}^*$ which, comparing equations (5) and (1), can be seen as the pseudo-example $(x_i, \mathbf{y})$ that violates its margin maximally. Instead of (5) we can thus solve the multilabel with maximum gradient

$$\begin{aligned} \mathbf{y}^* &= \underset{\mathbf{y}}{\mathbf{argmax}}\ g_i^T \mu_i^*(\mathbf{y}) \\ &= \underset{\mathbf{y}}{\mathbf{argmax}} \sum_{e \in E} g(i, e, \mathbf{y}_e)\mu(i, e, \mathbf{y}_e) \end{aligned} \tag{6}$$

and return the corresponding vertex $\mu^* = \mu(\mathbf{y}^*)$ of the marginal dual polytope. The problem (6) is readily seen as an inference problem on the Markov network $G$: one must find the configuration $\mathbf{y}^*$ that maximizes the sum of the 'edge potentials' $g(i, e, \mathbf{y}_e)\mu(i, e, \mathbf{y}_e)$.

Inference on a general graph is known to be hard. However, for our purposes, an approximate solution suffices: within an iterative algorithm it does not pay to spend a lot of time looking for optimal ascent direction when a reasonable ascent direction can be found fast. Here we opt to use loopy belief propagation with early stopping: we only compute a few iterations (given by the user parameter $maxLBPiter$) of the inference before returning (row 3 in Algorithm 1).

In addition to being the workhorse for optimizing the classification model, loopy belief propagation is also used in the prediction phase to extract the model's prediction given the learned parameters.

**Algorithm 1** Conditional gradient inference for single example.

**Input:** Initial dual variable vector $\mu_i$, gradient $g_i$, a joint kernel block $K_{ii}$ for the subspace
**Output:** New values for dual variables $\mu_i$.

1: $iter = 0$;
2: **while** $iter < maxcgiter$ **do**
3: $\quad \mu_i^* = feasibleDir(\mathbf{g}_i, E, maxLBPiter)$;
4: $\quad \tau = lineSearch(\mu_i^*, \mu_i, K_{ii})$;
5: $\quad$ **if** $\tau \leq 0$ **then**
6: $\quad\quad$ **break**; % no progress, stop
7: $\quad$ **else**
8: $\quad\quad \mu_i = \mu_i + \tau(\mu^* - \mu_i)$; % new solution
9: $\quad\quad g_i = g_i - \tau K_{ii}(\mu^* - \mu_i)$; % new gradient
10: $\quad$ **end if**
11: $\quad iter = iter + 1$;
12: **end while**

[1] We use the same probabilistic interpretation of dual variables as (Taskar et al., 2003).

[2] In the presence of ties, there is a set of vertices with optimum score; ties can be broken arbitrarily.

## 3 Kernels for drug-like molecules

Prediction of bioactivity is typically based on the physico-chemical and geometric properties of the molecules. Kernels computed from the structured representation of molecules extend the scope of the traditional approaches by allowing complex derived features to be used while avoiding excessive computational cost (Ralaivola et al., 2005). In this section, we will review the main approaches to construct a graph kernel for classification of drug-like molecules.

### 3.1 Walk Kernel

The classic way to represent the structure of a molecule is to use an undirected labeled graph $G = (V, E)$, where vertices $V = \{v_1, v_2, \ldots, v_n\}$ corresponds to the atoms and edges $E = \{e_1, e_2, \ldots, e_m\}$ to the covalent bonds. Vertex labels correspond to atom types (e.g. "oxygen", "carbon", etc.), and edge labels correspond to bond types (e.g. "single", "double", "aromatic", etc.). The $n \times n$ adjacency matrix $E$ of graph $G$ is defined such that its $(i, j)$'th entry $E_{ij}$ equals to one if and only if there is an edge between vertices $v_i$ and $v_j$.

Walk kernels (Kashima et al., 2003; Gärtner, 2003) compute the sum of matching walks in a pair of graphs. The contribution of each matching walk is downscaled exponentially according to its length. A *walk* of length $m$ in a graph $G$ is denoted by $w = \{v_1, v_2, ..., v_m\}$ such that for $i = 1, 2, ..., m-1$ there exists an edge for each pair of vertices $(v_i, v_{i+1})$.

A direct product graph between two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is denoted by $G_\times(G_1, G_2)$. Vertices of a product graph $G_\times(G_1, G_2)$ are defined as

$$V_\times(G_1, G_2) = \{(v_1, v_2) \in V_1 \times V_2, \; label(v_1) = label(v_2)\},$$

and edges are defined as

$$E_\times(G_1, G_2) = \{((v_1, v_2), (u_1, u_2)) \in V_\times \times V_\times, \; (v_1, u_1) \in E_1 \wedge (v_2, u_2) \in E_2\}.$$

The walk kernel can be defined as

$$K_{wk}(G_1, G_2) = \sum_{i,j=1}^{|v_\times|} \left[ \sum_{n=0}^{\infty} \lambda^n E_\times^n \right]_{ij},$$

where $v_\times$ is the vertex in product graph, $\lambda^n$ is the positive downscaling factor that is strictly less than one and $n$ is the length of walk.

Since longer walks are downscaled by $\lambda^n$, the contribution of longer walks are ofter negligible. Therefore, we consider finite-length walk kernel where only walks of length $p$ are explicitly constructed:

$$K_{wk_p}(G_1, G_2) = \sum_{v_i \in V_\times} D_p(v_i),$$

where $D_p(v_i)$ is calculated in a dynamic programming fashion by

$$D_0(v_i) = 1,$$
$$D_n(v_i) = \sum_{v_i, v_j \in E_\times} D_{n-1}(v_j).$$

### 3.2 Weighted decomposition kernel

The weighted decomposition kernel is an extension of the substructure kernel by weighting identical parts in a pair of graphs based on contextual information (Ceroni et al., 2007).

A weighted decomposition kernel on a labeled graph $G$ is based on a decomposition $D_r(G) = \{(s, z) : s \in V, z = \mathcal{N}_r(s)\}$, where $\mathcal{N}_r(s)$ is the neighborhood subgraph of radius $r$ of vertex $s$. The $s$ is called *selector*, and $z$ is the subgraph around it called *contextor*. A kernel function between two graphs $G_1$ and $G_2$ is defined as

$$K_{wdk}(G_1, G_2) = \sum_{\substack{v,z \in D_r(G_1) \\ v',z' \in D_r(G_2)}} [\![v = v']\!] K_s(z, z'),$$

where $K_s(z, z')$ is the kernel function between a pair of contextors. The function $K_s(z, z')$ uses the subgraph histogram intersection kernel discarding subgraph structure information defined as

$$K_s(z, z') = \sum_{l \in L} K_r(z, z'),$$
$$K_r(z, z') = \sum_{j=1}^{m_l} \min\{p_l(j), p'_l(j)\},$$

where $L$ is total number of attributes labeled on each vertex, $m_l$ is the number of possible values of the $l$'th property, and $p_l(j)$, $p'_l(j)$ are the observed frequencies of value $j$ for $l$'th attribute for subgraphs $z$ and $z'$, respectively.

### 3.3 Molecular fingerprints and the Tanimoto kernel

Molecular fingerprints are designed to encode a molecular structure into a fixed width binary bit vector that represents the presence or absence of substructures or fragments in the molecule. Molecular fingerprints are extensively used in chemical informatics.

There are two main types of fingerprints. *Hash fingerprints* enumerate all linear fragments of length $n$ in a molecule. Parameter $n$ is usually bounded from three to seven. A hash function assigns each fragment a hash value, which determines its position in descriptor space.

Another major fingerprint type is *substructure keys*, which is based on a pattern matching of a molecular structure to a set of pre-defined substructures. Each substructure becomes a key and has a fixed position in descriptor space. These substructures are considered to be independent functional units identified by domain experts as prior knowledge.

Once the molecules have been represented as fingerprints, the Tanimoto kernel (Ralaivola et al., 2005) is usually employed to measure the similarity between a pair of molecules. Given two molecular fingerprints $fp_1$ and $fp_2$, the Tanimoto kernel is defined as

$$K_{tk}(fp_1, fp_2) = \frac{N_{fp_1,fp_2}}{N_{fp_1} + N_{fp_2} - N_{fp_1,fp_2}},$$

where $N_{fp_1}$ is the number of 1-bits in fingerprint $fp_1$, $N_{fp_2}$ is the number of 1-bits in fingerprint $fp_2$, and $N_{fp_1,fp_2}$ is the number of 1-bits in both of the fingerprints.

## 4 Experiments

### 4.1 NCI-Cancer dataset

In this paper we use the NCI-Cancer dataset obtained through PubChem Bioassay[3] (Wang et al., 2009) data repository. The dataset initiated by National Cancer Institute and National Institutes of Health (NCI/NIH) contains bioactivity information of large number of molecules against several human cancer cell lines in 9 different tissue types, including leukemia, melanoma and cancers of the lung, colon, brain, ovary, breast, prostate, and kidney. For each molecule tested against a certain cell line, the dataset provides the bioactivity outcome that we use as the classes (active, inactive).

However, molecular activity data are highly biased over the cell lines. Figure 1 shows the molecular activity distribution over all 59 cell lines. Most of the molecules are inactive in all cell lines, while a relatively large proportion of molecules are active against almost all cell lines, which can be taken as toxics. These molecules are less likely to be potential drug candidates than the ones in the middle part of the histogram.

In order to circumvent the skewness and to concentrate on the most interesting molecules, we adopted the preprocessing suggested in (Shivakumar and Krauthammer, 2009), and selected molecules that are active in more than 10 cell lines and inactive in more than 10 cell lines. As a result, 544 molecules remained and were employed in our experiments.

### 4.2 Experiment setup and measures of success

To circumvent the skewness of the multilabel distribution, we use the following stratified cross-validation scheme to compare the methods: we divide examples into pools by the number of cell lines each molecule is active in (c.f. Figure 1). Then, we divide each pool into five

[3] http://pubchem.ncbi.nlm.nih.gov

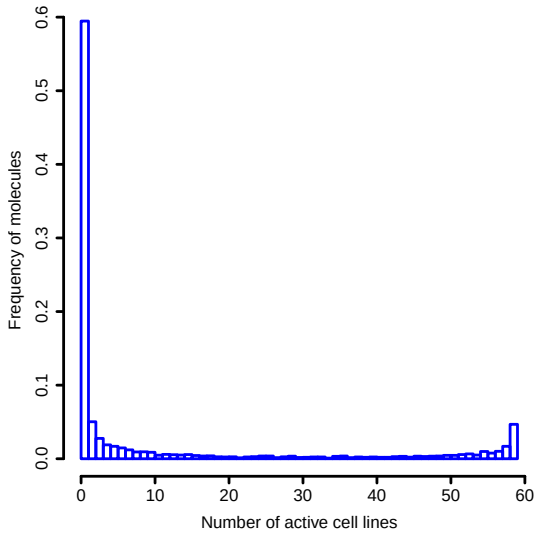


Figure 1: Skewness of the multilabel distribution.

folds and finally merge the corresponding pool-specific folds into five global folds.

To compare the performace of SVM and MMCRF in the multilabel prediction tasks we use microlabel accuracy and microlabel F1 score: we pool together individual microlabel predictions over all examples and all cell lines, and count accuracy and F1 from the pool.

### 4.3 Markov network generation for cancer cell lines

There are many ways one can build the Markov network in between the cell lines to be used as input to the MMCRF algorithm. For the dataset used in this paper, a large set of auxiliary data is available on the cancer cell lines from the NCI database[4]. Based on preliminary tests we opted to use RNA Radiation Array data. The basic approach is to construct from this data a correlation matrix between the pairs of cell lines and extract the Markov network from the matrix by favoring high-valued pairs. The following methods of network extraction were considered:

**SPT.**Maximum weight spanning tree. Take the minimum number of edges that make a connected network whilst maximizing the edge weights.

**CTh.**Correlation thresholding. Take all edges that exceed fixed threshold. This approach typically generates a general non-tree graph.

**Rnd.**Random graph. Draw edges uniformly at random.

In our experiments, the spanning tree approach on RNA radiation array data turned out to be best approach on average (Table 1). Suprisingly, correlation thresholding fails to meet the accuracy of random graph. However, the predictive performance of MMCRF surpasses SVM (c.f. Table 2) regardless of the network generation method.

Table 1: MMCRF Accuracy and F1 score with different Markov network extraction methods

| | **RNA Rad. array** | | |
|---|---|---|---|
| | ***SPT*** | ***CTh*** | ***Rnd*** |
| **Accuracy** | **67.6%** | 65.1% | 66.3% |
| **F1 Score** | **56.2%** | 52.8% | 53.5% |

Table 2: Accuracies and microlabel F1 scores from different kernels in MMCRF and SVM.

| ***Methods*** | ***Accuracy*** | ***F1 score*** |
|---|---|---|
| SVM + WK | 64.6% | 49.0% |
| SVM + WDK | 63.9% | 51.6% |
| SVM + Tanimoto | 64.1% | 52.7% |
| MMCRF + Tanimoto | **67.6%** | **56.2%** |

### 4.4 Effect of molecule kernels

We conducted experiments to compare the effect of various kernels, as well as the performances of support vector machine (SVM) and MMCRF. We used the SVM implementation of the LibSVM software package written in C++[5]. We tested SVM with different margin $C$ parameters, relative hard margin ($C = 100$) emerging

[4]http://discover.nci.nih.gov/cellminer/home.do

[5]http://www.csie.ntu.edu.tw/ cjlin/libsvm/

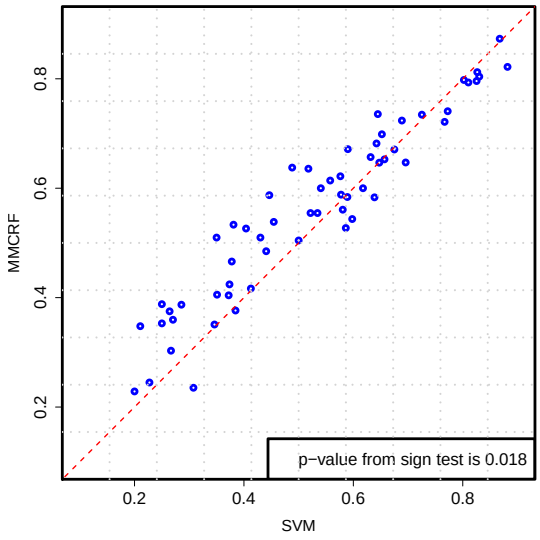


Figure 2: Microlabel F1 score of MMCRF against SVM classifier in each cell line.

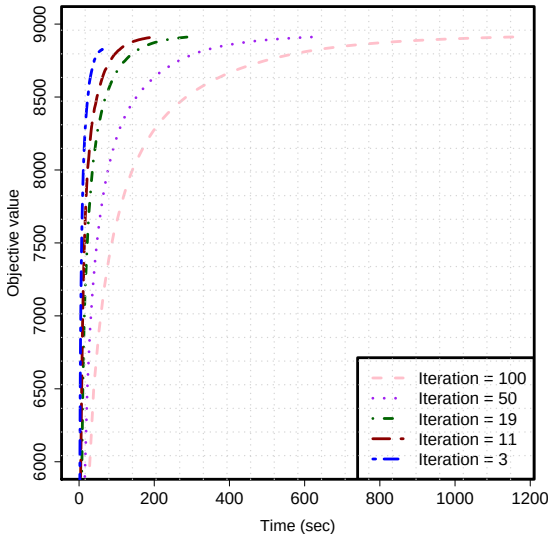


Figure 3: Effect of loopy belief propagation iteration.

as the value used in subsequent experiments. The same value was used for MMCRF classifier as well.

For the three kernel methods, walk kernel (WK) was constructed using parameters $\lambda = 0.1$ and $p = 6$ as recommended in (Gärtner, 2003). The Weighted decomposition kernel (WDK) used context radius $r = 3$ as in (Ceroni et al., 2007), and a single attribute (atom type) was sufficient to give the best performance. We also used hash fragments as molecular fingerprints generated by OpenBabel[6] (using default value $n = 6$ for linear structure length), which is a chemical toolbox available in public domain. All kernels were normalized.

In Table 2, we report overall accuracies and microlabel F1 scores using SVM with different kernel methods. The kernels achieve almost the same accuracy within SVM, while Tanimoto kernel is slightly better than others in microlabel F1 score. We further compared MMCRF and SVM classifiers with Tanimoto kernel. MMCRF turned out to outperform SVM in both overall accuracy and microlabel F1 score.

Figure 2 gives the F1 score in each cell line from MMCRF classifier against SVM classifier in the same experiment. Points above the diagonal line correspond to improvements in F1 scores by MMCRF classifier. MMCRF improves microlabel F1 scores of 39 out of 59 cell lines with sign test giving the $p$-value of 0.018. The statistics for accuracy were similar (data not shown).

[6]http://openbabel.org

### 4.5 Effect of loopy belief propagation

Finally, we tested different loopy belief propagation iteration parameters to see the their effects on convergence (Figure 3). The best loopy belief propagation iteration limit turned out to be $maxLBPiter = 11$, Smaller values were not sufficient for MMCRF to reach a global optimum, while larger values caused the convergence to need more time. The optimal value turned out to be close to the diameter of the Markov network (10 in this case), indicating that propagation of messages through the whole network is required for best performance.

## 5 Conclusions

We presented a multilabel classification approach to drug activity classification using the Max-Margin Conditional Random Field algorithm. In experiments against a large set of cancer lines the method significantly outperformed SVM.

## Acknowledgements

This work was financially supported by Academy of Finland grant 118653 (ALGODAN) and in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. This publication only reflects the authors' views.

## References

L. Bernazzani, C. Duce, A. Micheli, V. Mollica, A. Sperduti, A. Starita, and M.R. Tine. 2006. Predicting physical-chemical properties of compounds from molecular structures by recursive neural networks. *J. Chem. Inf. Model.*, 46:2030–2042.

E. Byvatov, U. Fechner, J. Sadowski, and G. Schneider. 2003. Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. *J. Chem. Inf. Comput. Sci.*, 43:1882–1889.

A. Ceroni, F. Costa, and P. Frasconi. 2007. Classification of small molecules by two- and three-dimensional decomposition kernels. *Bioinformatics*, 23:2038–2045.

P. de Waal and L. van der Gaag. 2007. Inference and Learning in Multi-dimensional Bayesian Network Classifiers. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 501–511.

T. Gärtner. 2003. A survey of kernels for structured data. *SIGKDD Explor. Newsl.*, 5(1):49–58.

H. Kashima, K. Tsuda, and A. Inokuchi. 2003. Marginalized kernels between labeled graphs. In *Proceedings of the* $20^{\text{th}}$ *International Conference on Machine Learning (ICML)*, Washington, DC, United States.

R. King, S. Muggleton, A. Srinivasan, and M. Sternberg. 1996. Structure-activity relationships derived by machine learning: the use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *PNAS*, 93:438–442.

L. Ralaivola, S. Swamidass, H. Saigo, and P. Baldi. 2005. Graph kernels for chemical informatics. *Neural Networks*, 18:1093–1110.

JD Rodriguez and J.A. Lozano. 2008. Multi-objective learning of multi-dimensional bayesian classifiers. In *Eighth International Conference on Hybrid Intelligent Systems, 2008. HIS'08*, pages 501–506.

J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. 2006. Kernel-Based Learning of Hierarchical Multilabel Classification Models. *The Journal of Machine Learning Research*, 7:1601–1626.

J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. 2007. Efficient algorithms for max-margin structured classification. *Predicting Structured Data*, pages 105–129.

P. Shivakumar and M. Krauthammer. 2009. Structural similarity assessment for drug sensitivity prediction in cancer. *Bioinformatics*, 10:S17.

C.N. Silla and A.A. Freitas. 2010. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, pages 1–42.

S.J. Swamidass, J. Chen, J. Bruand, P. Phung, L. Ralaivola, and P. Baldi. 2005. Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics*, 21:359–368.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin markov networks. In *Neural Information Processing Systems 2003*.

M. Trotter, M. Buxton, and S. Holden. 2001. Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Comp. and Chem.*, 26:1–20.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y.n Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. 21th International Conference on Machine Learning*, pages 823–830.

M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. 2005. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717.

Y. Wang, E. Bolton, S. Dracheva, K. Karapetyan, B.A. Shoemaker, T.O. Suzek, J. Wang, J. Xiao, J. Zhang, and S.H. Bryant. 2009. An overview of the pubchem bioassay resource. *Nucleic Acids Research*, 38:D255–D266.

V. Zernov, K. Balakin, A. Ivaschenko, N. Savchuk, and I. Pletnev. 2003. Drug discovery using support vector machines. The case studies of drug-likeness, agrochemical-likeness, and enzyme inhibition predictions. *J. Chem. Inf. Comput. Sci.*, 43:2048–2056.