Multilabel Classification of Drug-like Molecules

Hongyu Su

Helsinki April 30, 2015

M.Sc. Thesis

Master's Degree Programme in Bioinformatics

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Laitos — Institution — Department Department of Computer Science

Tekijä — Författare — Author

Hongyu Su

Työn nimi — Arbetets titel — Title

Multilabel Classification of Drug-like Molecules

Oppiaine — Läroämne — Subject

Bioinformatics

Työn laji — Arbetets art — Level M.Sc. Thesis	Aika — Datum — Month and year April 30, 2015	Sivumäärä — Sidoantal — Number of pages 0 pages + 0 appendices

Tiivistelmä — Referat — Abstract

The increasing availability of large data repositories of chemical compounds greatly facilitates the drug discovery process and offers both challenges and opportunities to machine learning. The task studied in the thesis is to predict the bioactivity of the molecules so that the costly *in vitro* and *in vivo* drug testing can focus on the few most promising drug candidates.

Molecular classification is originated in the early research of structure activity relationships. It is based on the assumption that chemical or biological properties directly relates to geometric or physiological structures. After that, a number of methods have been proposed and extensively studied. Support vector machine with kernel methods emerged as an computationally effective way to handle non-linear properties of molecules. In particular, they offer promising results on molecular classification task and are regarded as the state-of-the-art. However, the binary classification methods which predict the bioactivity against a single target at a time are not sufficient, especially when a large number of putative targets are associated to each molecule.

In this thesis, we introduce the first multilabel learning approach for molecular classification task using structured output prediction. The relationships between multiple targets are modeled by a Markov network. Max-margin learning is used for separating the correct multilabel from the incorrect ones. The experiments are mostly based on NCI-cancer dataset which consists of molecular data for 60 cancer cell line targets. The results show that the multilabel classification approach outperforms the state-of-the-art support vector machine approach in a statistically significant manner.

ACM Computing Classification System (CCS):
A. General Literature,
  A.1 Introductory and Survey
I. Computing Methodologies,
  I.6 Simulation and Modeling,
    I.6.3 Applications
    I.6.4 Model Validation and Analysis
    I.6.5 Model Development
    I.6.6 Simulation Output Analysis
J. Computer Applications,
  J.3 Life and Medical Sciences

Avainsanat — Nyckelord — Keywords

kernel methods, structured outputs, drug discovery

Säilytyspaikka — Förvaringsställe — Where deposited

Kumpula Science Library, serial number C-

Muita tietoja — övriga uppgifter — Additional information

Acknowledgement

I could not have my thesis without the guidance of my supervisors. First and foremost, I would like to deeply thank my supervisor Professor Juho Rousu, who lets me freely explore and find my inspiration while timely putting me back to the right track. The encouragements and advise from him are indispensable during the whole procedure. I am also grateful to my supervisor Docent Raimo Ketola for his patience and expertise in reading and commenting the thesis.

I am indebted to Markus Heinonen, my tutor, who is always there offering ideas and dealing with my stupid questions. I would also like to thank the people in CSBB group, for the invaluable discussions with Esa Pitkänen and the kind help from Katja Astikainen. The group meetings really help keeping me in track with ideas in related research areas. I should also thank all my friends in Department of Computer Science, to name but a few: Francois, Fang, Janne, Laura, Lu, Pasi and Sebastien for the coffees, lunches, talks and all the happy time that we shared together. I would like to thank all lecturers and professors in the MBI program for teaching me so many things, and all my classmates for enjoying or suffering in every lectures, wet labs and exams together.

I would like to acknowledge the funding provided by Academy of Finland grant 118653 (ALGODAN).

Finally, I thank my parents for being there and supporting me all the time.

Helsinki, August 2010
Hongyu Su

Contents

1 Introduction 1

2 Support vector machine and kernel methods 3

2.1 Support vector machine basis . . . 3

2.2 Margin maximization . . . 3

2.3 Soft margin optimization . . . 7

2.4 Non-linear SVM . . . 10

2.5 Kernel methods . . . 11

2.6 Multiclass SVM . . . 13

2.7 Multilabel classification . . . 15

2.7.1 Max-margin conditional random field (MMCRF) . . . 16

2.7.2 Max-margin learning . . . 17

2.7.3 Marginal dual problem . . . 18

2.7.4 Optimization and inference . . . 19

3 Molecular representations 20

3.1 Molecular graph . . . 20

3.2 Reduced graph . . . 21

3.2.1 Topological pharmacophore reduction . . . 22

3.2.2 Functional group reduction . . . 22

3.3 Molecular descriptors . . . 24

3.3.1 Substructure key . . . 24

3.3.2 Hash fragments . . . 25

3.3.3 Pharmacophore fingerprint . . . 26

4 Kernels for drug-like molecules 27

4.1 Notation . . . 27

4.2 Walk kernel . . . 27

4.3 Weighted decomposition kernel . . . 33

4.4 Tanimoto kernel . . . 34

4.5 MinMax kernel . . . 36

4.6 Hybrid kernel . . . 37

4.7 Tree kernel . . . 37

5 Experimental data **39**

5.1 Mutag dataset 39

5.2 NCI-cancer datasets 40

5.3 Auxiliary data and Markov network 46

6 Experimental results **49**

6.1 Measurement of success 49

6.1.1 Statistical measurement 49

6.1.2 Validation methods 51

6.2 Experimental setup 51

6.3 Binary classification 52

6.3.1 Mutag dataset 52

6.3.2 NCI-cancer dataset 53

6.4 Multilabel structural classification 55

6.4.1 Effect of Markov network 56

6.4.2 Effect of dataset version 56

6.4.3 Agreements of predictions from SVM and MMCRF 57

6.4.4 Effect of loopy belief propagation 58

6.4.5 Computation time 61

7 Conclusions **64**

1 Introduction

Drug discovery is facilitated by big data repositories of chemical compounds from ultra-high-throughput screening techniques, where large number of molecules are tested and classified based on their activities against given targets. The increasing availabilities of data offers both challenges and opportunities to machine learning. The task in the thesis is to infer chemical or biological properties of molecules from structural representations in different dimensions. In particular, an accuracy *in silico* model will filter out a large number of unqualified molecules, and the costly preclinical trial can focus on few potential drug candidates [?]. These models are based on the assumption that molecules with similar geometric or physiologic properties will also have similar chemical and biological characteristics [?].

Various methods have been proposed and extensively studied in order to deal with molecular classification task, including the early correlation analysis of quantitative structure activity relationship (QSAR) [?], the recent inductive logic programming approach [?], artificial neural networks methods [?, ?], Bayesian classification approaches [?, ?] and decision tree methods [?].

Support vector machine (SVM) and kernel methods [?] emerged during the last decade and became important prediction methods in machine learning. They are suitable for tackling the non-linear properties of structured data of various kinds. The advantage of the methods is that the learning machine does not necessarily have to access high dimensional feature space when it interacts with objects in the dataset. The challenges for kernel methods is to construct a good kernel for a given task. In molecule classification problem, the kernels correspond to functions capable of measuring similarity between molecules based on different molecular representations. One of the first successful applications of kernel methods in molecule classification is often known as walk kernel continuously developed in [?, ?, ?]. After that, a variety of kernel methods were developed and applied in this area, including kernels based on molecular fingerprints [?], kernels derived from differet string representations [?], and kernel considering local substructures [?, ?] or three-dimensional substructures [?]. It should be noted that subgraph kernels were proven hard to calculate [?], as they are NP-hard of subgraph isomorphism problems in nature. However, subgraph kernels, which are restricted to several salient subgraphs and applied to moderate-sized datasets, can still achieve promising results. These methods are usually known as reduced graph approaches [?].

Although methods, that focus on a single target variable at a time commonly known as binary classification, became the mainstream methods for molecular classification task, they are not perfectly suitable for drug screening scenario where a number of targets need to be evaluated at the same time. For example more than 60 target values are connected to each molecule from the anticancer therapy screening experiments [?]. The corresponding machine learning problem is known as multilabel classification, and can be roughly tackled by a collective binary classification procedures. However, they neglect the relationships between target variables.

We addressed the multi-target scenario of molecular data by employing a multilabel structural learning approach. Our approach belongs to a structural output leaning family [?, ?, ?, ?], and can be taken as an instantiation of the algorithm proposed in [?, ?] known as max-margin conditional random field learning (MMCRF). The method is a combination of the kernel method, the SVM liked optimization and the probabilistic graphical model. It arranges the multiple target variables into a Markov network, takes as input the kernel of molecules, and uses the discriminative max-margin optimization approach for learning parameters. MMCRF uses belief propagation inference over Markov network to learn the model and to construct predictions. From our experiments, our multilabel structural learning approach clearly outperforms traditional binary classification approach conducted by SVM.

In this thesis, we will describe the definitions and the algorithms of both binary classification by SVM and the multilabel structural classification by MMCRF. Then, we will explain different ways to represent and describe molecules. After that, we will review popular kernel methods for molecular classification. Finally, we will show from our experiments the best kernel method for binary classification and the improvements we got by employing multilabel structural classification approach.

2 Support vector machine and kernel methods

2.1 Support vector machine basis

Support vector machine (SVM) is a supervised learning algorithm that can be used in classification and regression. SVM was first introduced in [?]. A detailed derivation of support vector machine can be found in the book [?]. Other information about support vector machine are provided in [?, ?, ?, ?]. In the recent years, many new methods related to SVM have been developed and successfully applied to many challenging problems in many areas [?, ?].

We begin our discussion of the algorithm based on *binary classification* problem, where the training dataset $\mathcal{S}$ consists of a series of n objects $\mathcal{X} = \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$ together with a series of labels $\mathcal{Y} = y_1, y_2, \cdots, y_n$ associated with the objects. Objects are usually formalized as vectors indicated by $\mathcal{X} \in \mathbb{R}^n$ and each object is classified into one of two classes indicated by $y_i \in \{+1, -1\}$. Support vector machine tries to separate two classes of points by learning from $\mathcal{S}$ a function

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{1}$$

with $\mathbf{w} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}$. The function will give a label $+1$ to $\mathbf{x} \in \mathcal{X}$ when $f(\mathbf{x}) \geq 0$, and a label -1 to $\mathbf{x} \in \mathcal{X}$ when $f(\mathbf{x}) < 0$.

2.2 Margin maximization

Given a candidate function that takes the form of (1), one can check each observation $(\mathbf{x}_i, y_i)$ in training data set $\mathcal{S}$ to see whether it is correctly classified, which happens when $y_i f(\mathbf{x}_i) \geq 0$. Intuitively, a good function is one with minimum number of misclassified observations in training data set $\mathcal{S}$, denoted by $y_i f(\mathbf{x}_i) < 0$. This is commonly known as *empirical risk minimization*. An example of empirical risk minimization is shown in Figure 1.

Linear perceptron (e.g. [?]) is an algorithm that is guaranteed to find a solution in a finite number of steps, with regards to minimizing empirical risk. However, the solution from linear perceptron algorithm is not unique when there exists more than one function to separate two classes of points that have the same empirical risk, shown in Figure 2. The solutions are greatly affected by the initial values of $\mathbf{w}$ and b, as well as the order to which observations in $\mathcal{S}$ are presented to perceptron.

When multiple solutions exist, SVM employs the concept of *margin maximization* to work out this problem. *Margin* is defined as the distance between the decision boundary and the the closest data point. Margin maximization is to select a hyperplane with largest distances towards the closest data points, shown in Figure 3. Margin maximization can lead to a good generalization that has small error on unseen data points.

Assume data are linearly separable in training data set $\mathcal{S}$. We can always select a hyperplane for each of two classes and make sure there are no data points between

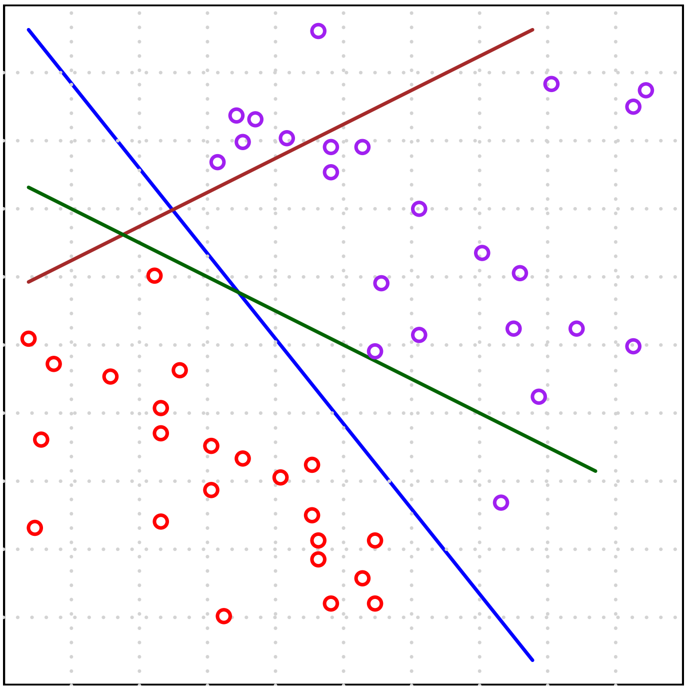


Figure 1: Empirical risk. Points in 2D space of two classes (red and purple) are classified by three linear classifiers (blue, green and dark red lines). The measure of success is to minimize empirical risk with respect to the number of misclassified points. The dark red lines leads five points to be misclassified, one point is misclassified by the green line, and the blue line has all points correctly classified. Therefore the blue line is the best separator in this case.

them. Decision boundary locates in the middle of the two hyperplanes. We try to maximize the distance between the two hyperplanes which also maximizes the distance between the decision boundary and the closest data points, the margin of SVM. The distance between two hyperplane is found to be $\frac{2}{||\mathbf{w}||}$. Hence, margin maximization is to minimize $||\mathbf{w}||$ with the constraint that each data point is correctly classified. This is commonly known as *hard margin SVM* and can be defined as the following optimization problem

$$(\mathbf{w}, b) = \underset{\mathbf{w},b}{\mathbf{argmin}} \frac{1}{2}||\mathbf{w}||^2, \tag{2}$$

$$\text{s.t. } y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1, \tag{3}$$

$$1 \leq i \leq n.$$

This is an example of *quadratic programming* problem, which is to minimize a quadratic function with constraints of a set of linear equalities. SVM is not directly solved by (2), but by *dual optimization.* We introduce a set of *Lagrange multipliers* $\alpha = (\alpha_i, \cdots, \alpha_n)$ with $\alpha_i \geq 0$ for all i, one for each constraint of (3). It

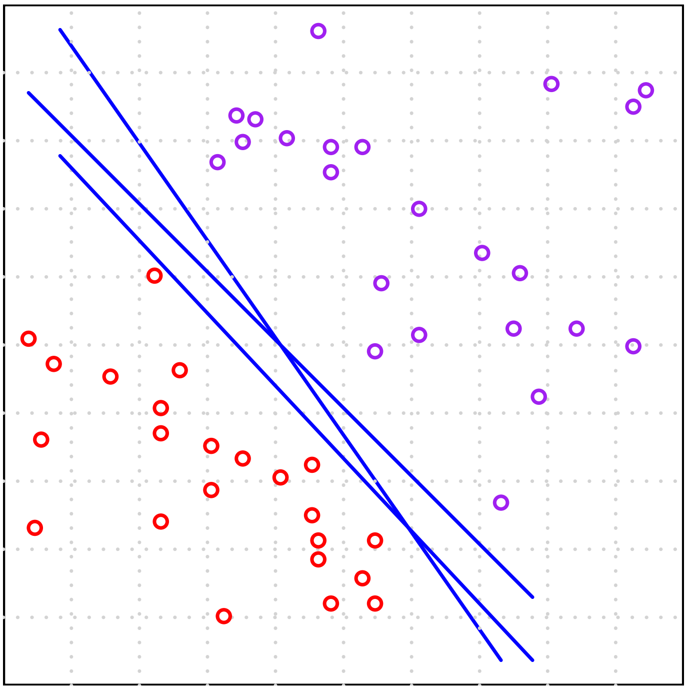


Figure 2: Linear perceptron. Points in 2D space are classified by three hyperplanes. Data are linearly separable and all of three hyperplane can correctly classify the data points. Therefore they are equivalent classifiers with respect to empirical risk minimization. The empirical risk principle does not uniquely define the solution in this case.

results in the *primal representation* of Lagrange function, where we minimize

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1). \tag{4}$$

The Lagrange function can be interpreted as minimization with respect to $\mathbf{w}$ and b and maximization with respect to α_i. Taking the partial derivatives of $\mathcal{L}(\mathbf{w}, b, \alpha)$ with respect to $\mathbf{w}$ and b results in following two constraints:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0,$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0.$$

Inserting the constraints from partial derivatives back to the primal (4) gives the *dual representation*, in which we maximize

$$\tilde{\mathcal{L}}(\alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j), \tag{5}$$

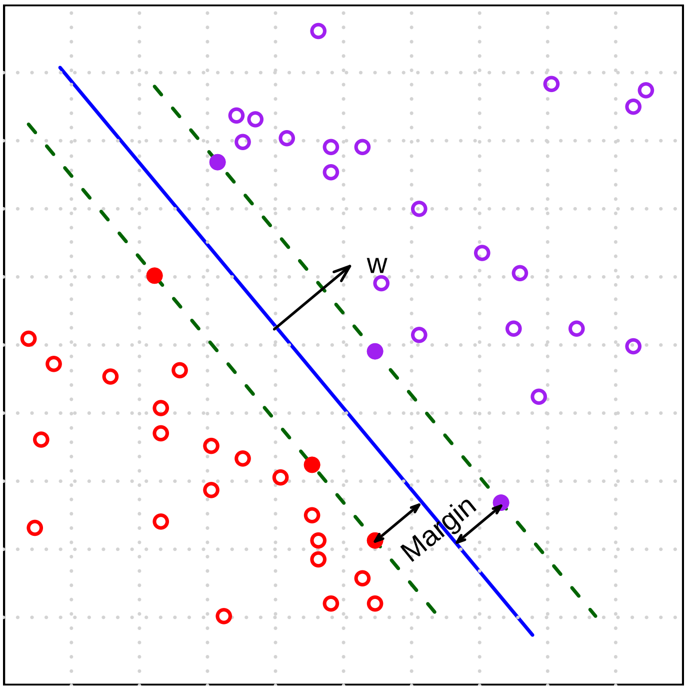


Figure 3: Margin maximization. Data are linearly separable. A hyperplane defined by parameters $\mathbf{w}$ and b is shown as a blue line. The distance between the closest data points and the blue line is defined as margin. The green dashed lines are margin hyperplanes. Data points on margin hyperplanes are the so called support vectors.

where $(\mathbf{x}_i \cdot \mathbf{x}_j)$ denotes the inner product of observations $\mathbf{x}_i$ and $\mathbf{x}_j$. It is subject to the positivity constraint $\alpha_i \geq 0$ for $1 \leq i \leq n$ and the equality constraint $\sum_{i=1}^{n} \alpha_i y_i = 0$.

Predictions can be made by evaluating the sign of (1). Using learned parameters α to substitute $\mathbf{w}$ results in

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + \mathbf{b}. \tag{6}$$

According to Karush-Kuhn Tucker (KKT) conditions [**?**], among observations $(\mathbf{x}_i, y_i)$ in training data set $\mathcal{S}$, ones that lies on the correct side of the margin hyperplane will have $\alpha_i = 0$. They play no role in making predications of new data points, seen from (6). The remaining observations will have $\alpha_i > 0$. They locate on the margin hyperplane and satisfy $y_i f(\mathbf{x}_i) = 1$. These observations are known as *support vectors*, and are important in SVM theory. After one learns a model from training data set, only a small proportion of examples need to be retained while others can be discarded.

Once we solve the quadratic problem and find the values of α, we can determine the parameter b by using the equality $y_i f(\mathbf{x}_i) = 1$ of any support vector $(\mathbf{x}_i, y_i)$.

Combining with (6) leads to

$$y_j \left(\sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}_j) + b \right) = 1. \tag{7}$$

Since non-*support vectors* will have $\alpha_i = 0$, we further rewrite the (7) with only *support vectors* as:

$$y_j \left(\sum_{m \in \mathcal{SV}} \alpha_m y_m (\mathbf{x}_m \cdot \mathbf{x}_j) + b \right) = 1, \tag{8}$$

where $\mathcal{SV}$ denote the index set of support vectors. Therefore, we can obtain the value of parameter b with an arbitrary support vector by:

$$b = y_i - \sum_{m \in \mathcal{SV}} \alpha_m y_m (\mathbf{x}_m \cdot \mathbf{x}_i)$$

For numerical stability [**?**], one common way is to take the average value of b obtained from each support vector by:

$$b = \frac{1}{\mathcal{N}_{\mathcal{SV}}} \sum_{i \in \mathcal{SV}} \left(y_i - \sum_{m \in \mathcal{SV}} \alpha_m y_m (\mathbf{x}_m \cdot \mathbf{x}_i) \right),$$

where $\mathcal{N}_{\mathcal{SV}}$ is the total number of support vectors.

2.3 Soft margin optimization

So far we have assumed that training data are linearly separable and hard margin SVM can exactly classify examples in two classes. In most real world cases data are not linearly separable. Training examples of two classes in feature space $\mathcal{X}$ may overlap with each other. It is sometimes possible to classify all examples with a non-linear separator. However, this may lead to loss of generality and poor predictions of unseen data points.

We need to modify hard margin SVM to allow some of training examples be misclassified, shown in Figure 4. We therefore introduce n new parameters $\xi = (\xi_1, \cdots, \xi_n)$ where $\xi_i \geq 0$, with one for each training data points. Parameters ξ are known as *slack parameters*, and were first introduced in [**?**]. They are defined so that $\xi_i = 0$ corresponds to data points that are outside or on the margin boundary, and $\xi_i = |y_i - f(\mathbf{x}_i)|$ for others. More specifically, data points with $0 < \xi_i < 1$ locate between margin boundary and decision boundary, ones with $\xi_i = 1$ are on decision boundary, and ones with $\xi_i > 1$ are misclassified.

Intuitively, hard margin constraint being relaxed by slack parameters leads to a *soft margin*. The goal is therefore to maximize margin while penalize the data points

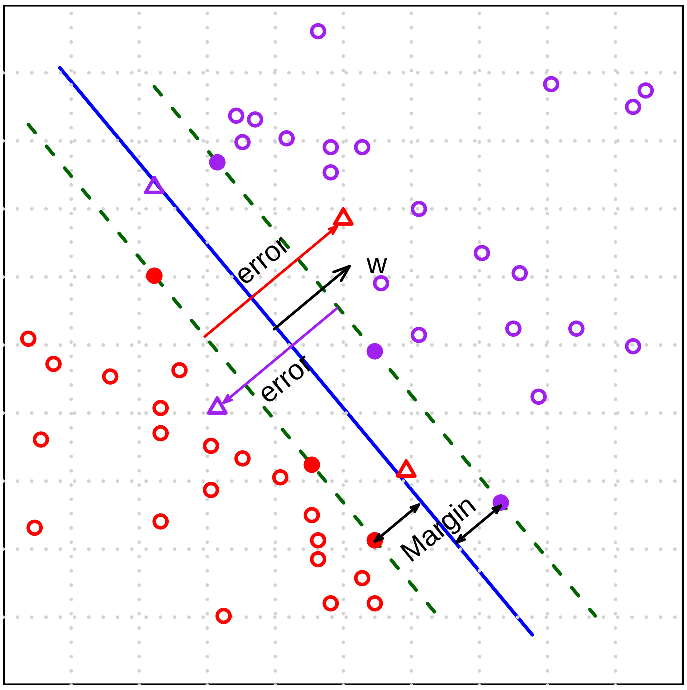


Figure 4: Soft margin SVM. The majority of data are linearly separable (circle points), with some of them overlaps (triangle points). A hyperplane is shown to classify points into two classes with respect to maximizing the margin. The optimization of soft margin SVM allows some points to be misclassified at the same time.

that locate on the wrong side of margin boundary. This is commonly known as *soft margin SVM* and can be defined as the following optimization problem:

$$(\mathbf{w}, b) = \underset{\mathbf{w},b}{\mathbf{argmin}} \left(\frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{m} \xi_i \right), \tag{9}$$

$$\text{s.t. } y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \geq 1 - \xi_i, \tag{10}$$

$$\xi_i \geq 0,\ 1 \leq i \leq n, \tag{11}$$

where $C > 0$ is the parameter that balances the margin maximization and penalties of misclassified examples (training errors). When $C \to \infty$, misclassifications are weighted so greatly that soft margin SVM defined in (9) goes back to the earlier hard margin SVM defined in (2).

This is also an example of quadratic programming problem. We process the problem with Lagrange multiplier by introducing $\alpha = (\alpha_i, \cdots, \alpha_n)$ with one for each constraint of (10), and $\beta = (\beta_i, \cdots, \beta_n)$ with one for each constraint of (11). It leads to primal form of Lagrange function, where we minimize

$$\mathcal{L}(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i. \tag{12}$$

Taking the partial derivatives of $\mathcal{L}(\mathbf{w}, b, \xi, \alpha, \beta)$ with respect to $\mathbf{w}$, b and ξ results in following constraints:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i = 0, \\
\frac{\partial \mathcal{L}}{\partial b} &= \sum_{i=1}^{n}\alpha_i y_i = 0, \\
\frac{\partial \mathcal{L}}{\partial \xi_i} &= C - \alpha_i - \beta_i = 0,\ 1 \le i \le n.
\end{aligned}$$

Plunging the constraints from partial derivatives back to primal defined in (12) and eliminating $\mathbf{w}$ and b gives the dual form, in which we maximize

$$\tilde{\mathcal{L}}(\alpha) = \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i \cdot \mathbf{x}_j), \tag{13}$$

$$\begin{aligned}
\text{s.t. } & 0 \le \alpha_i \le C, 1 \le i \le n, \\
& \sum_{i=1}^{n}\alpha_i y_i = 0.
\end{aligned}$$

This is almost identical to the dual form of hard margin SVM defined in (5), except subjecting to different constraints. Once α_i is obtained, another dual variable β can be found as:

$$\beta_i = C - \alpha_i,\ 1 \le i \le n.$$

Similarly, predictions for new data points are based on (6). Training examples with $\alpha_i = 0$ do not affect the decisions. Examples with $\alpha_i = C$ require $\beta_i = 0$ which leads to either $\xi_i = 0$ corresponding to being inside the margin, or $0 < \xi_i < 1$ corresponding to being between margin boundary and decision boundary, or $\xi_i \ge 1$ corresponding to being misclassified. According to KKT condition, examples with $0 < \alpha_i < C$ result in $\xi_i = 0$, which correspond to being exactly on the margin boundary. They are support vectors in soft margin SVM. A numeric stable solution for $\mathbf{b}$ is given by

$$b = \frac{1}{\mathcal{N}_{\mathcal{SV}'}}\sum_{i\in\mathcal{SV}'}\left(y_i - \sum_{m\in\mathcal{SV}'}\alpha_m y_m(\mathbf{x}_m \cdot \mathbf{x}_i)\right),$$

where $\mathcal{N}_{\mathcal{SV}'}$ is the total number of support vectors in soft margin SVM that have $0 < \alpha_i < C$.

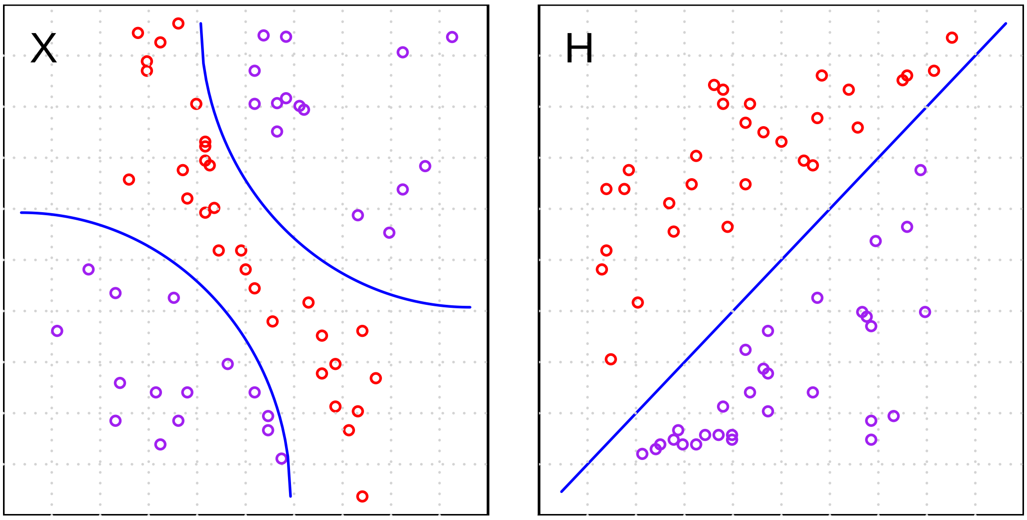


Figure 5: Non-linear SVM. Left, data points in 2D space are given in two classes. Even using soft-margin SVM with slack parameter ξ, it is not possible to classify the data points with a linear hyperplane. Right, non-linear SVM uses a feature mapping function φ to transform data points in input space to a high dimensional feature space $\mathcal{H}$. There may exist a linear separator in $\mathcal{H}$ that can classify the data points.

2.4 Non-linear SVM

In the previous cases, we have discussed the SVM algorithm in the scope that an optimal classification hyperplane exists for a linear separation of training data. However, the optimal separation surface are not simply linear in many cases. Even with slack parameters, it is not possible to separate the training data with a linear separator, as shown in Figure 5.

We can extend support vector machine to tackle the non-linear data by means of a feature mapping function $\varphi(\mathbf{x})$, which transforms the examples in input space $\mathcal{X}$ to a higher dimensional feature space $\mathcal{H}$, where training data is supposed to be classified by a linear separator. $\mathcal{H}$ is commonly known as a *Hilbert* space with finite or infinite dimensions. Mathematically, training dataset $\mathcal{S}$ consists of a series of n objects $\mathcal{X} = \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n \in \mathbb{R}^n$, where $\mathbf{x}_i$ is a n dimensional vector $\mathbf{x}_i = (x_1, x_2, \cdots, x_n)$. A feature mapping function is comprised by a set of functions $\varphi = (\varphi_1, \varphi_2, \cdots, \varphi_k)$ that transfers each example $\mathbf{x}_i$ to $\varphi(\mathbf{x}_i)$, defined as

$$\mathbf{x}_i = (x_1, x_2, \cdots, x_n) \xrightarrow{\varphi} \varphi(\mathbf{x}_i) = (\varphi_1, \varphi_2, \cdots, \varphi_k).$$

Note $\mathbf{x}_i$ and φ can be in different dimensions, finite or infinite. As a result, we obtain a set of objects in high dimensional feature space $\varphi(\mathcal{X}) = (\varphi(\mathbf{x}_i), \varphi(\mathbf{x}_2), \cdots, \varphi(\mathbf{x}_n)) \in \mathcal{H}$. Intuitively, even if training data in the input space $\mathcal{X}$ are not linear separable, they may be separated by a hyperplane in high dimensional feature space $\mathcal{H}$ if there exists an appropriate feature mapping function $\varphi : \mathbf{x} \xrightarrow{\varphi} \varphi(\mathbf{x})$. The support vector machine with a non-linear feature mapping function is known as *non-linear SVM*, shown in Figure 5.

Substitute $\mathbf{x}$ by $\varphi(\mathbf{x})$ in equations that defines the soft-margin SVM, we can obtain the mathematical formulations of the non-linear SVM. Specifically, optimization problem given in primal form is to minimize

$$\mathcal{L}(\mathbf{w}, \mathbf{b}, \xi, \alpha, \beta) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}^T\varphi(\mathbf{x}_i) + \mathbf{b}) - 1 + \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i. \tag{14}$$

The corresponding dual form is given by maximizing

$$\tilde{\mathcal{L}}(\alpha) = \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\varphi(\mathbf{x}_i)\cdot\varphi(\mathbf{x}_j)). \tag{15}$$

Similarly, classification is done through the decision function give by

$$f(\mathbf{x}) = \sum_{i=1}^{n}\alpha_i y_i(\varphi(\mathbf{x}_i)\cdot\varphi(\mathbf{x})) + \mathbf{b}. \tag{16}$$

2.5 Kernel methods

Note that the problem of learning in non-linear separable data has been transformed to mapping data into feature space of higher dimension and performing a linear classification in the feature space. However, the dimensionality of feature space can be very high or even infinite and the feature mapping function can be quite complex. Therefore, we hope that we can avoid the direct inner product $\varphi(\mathbf{x}_i)\cdot\varphi(\mathbf{x}_j)$ in the feature space. This motivates the *kernel methods*.

A function $K : \mathcal{X} \times \mathcal{X} \xrightarrow{K} \mathbb{R}$ is called *positive definite kernel* if and only if it is symmetric $K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i)$ for any $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, and for any $C_i, C_j \geq 0$ it satisfies

$$\sum_{i=1}^{n}\sum_{i=1}^{n}C_iC_jK(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

It is also proved [?] that for any kernel function K defined on the input space $\mathcal{X}$, a mapping function $\varphi : \mathcal{X} \xrightarrow{\varphi} \mathcal{H}$ exists, which satisfies

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)\cdot\varphi(\mathbf{x}_j),\ \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}.$$

This property is very meaningful because we have to know neither the exact form of feature mapping function φ nor the explicitly representations of objects in high dimensional feature space $\varphi(\mathbf{x})$. Only inner product is sufficient and can be calculated in input space with a kernel function.

Replacing inner product $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$ with kernel function results in *kernel SVM.* The optimization problem in dual form is given by maximization of

$$\begin{aligned} \tilde{\mathcal{L}}(\alpha) = & \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \\ \text{s.t. } & \sum_{i=1}^{n} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, 1 \leq i \leq n. \end{aligned} \tag{17}$$

The decision function is given by

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + \mathbf{b}. \tag{18}$$

Kernel function takes pairs of examples from input space. Therefore it can take various form, as long as it satisfies positive definite conditions. Commonly used kernel functions include:

Linear Kernel is the inner product of two training examples commonly used as a test for non-linear problem and serves as the reference for improvements achieved by other kernel functions, defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j).$$

Polynomial Kernel is the simplest way to model the non-linear relationship which maps data into a higher dimensional space. The degree of the space is denoted by parameter d, and the parameter c decides the smoothness of the resulted kernel. The polynomial kernel is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d.$$

Radial Basis Function Kernel is a widely used kernel function whose shape is controlled by parameter γ, defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2),\ \gamma > 0.$$

Gaussian Radial Basis Function Kernel is another widely used kernel function which maps data into infinite dimension, defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma}).$$

Sigmoid Kernel corresponds to the most used transformation function in neural networks with a sigmoid shape, defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(a\mathbf{x}_i \cdot \mathbf{x}_j + b).$$

Additive Kernel follows one property of kernel function which is the summation of kernel functions is still a valid kernel function, defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \sum_m K_m(\mathbf{x}_i, \mathbf{x}_j).$$

Product Kernel follows another property of kernel function which is the tensor product of kernel functions is still a valid kernel function, defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \prod_m K_m(\mathbf{x}_i, \mathbf{x}_j).$$

Kernel normalization is independent of kernel types. It is defined as

$$\tilde{K}(\mathbf{x}_i, \mathbf{x}_j) = \frac{K(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{K(\mathbf{x}_i, \mathbf{x}_i)K(\mathbf{x}_j, \mathbf{x}_j)}}.$$

This normalization ensures that the diagonal values of kernel matrix are 1 and other values range from 0 to 1. It is quite important especially for generalized kernel functions, which are computed direct from objects other than feature representations. kernel normalization can avoid the biased results, for example bigger graph will result in large value in kernel matrix when calculating graph kernels.

2.6 Multiclass SVM

The support vector machine is specially designed for classification of data in two classes, or so called binary classification problem. Many real world problem consists of, however, more than two classes. For example, functions of a protein or an enzyme can possibly belong to several EC families[1]. Therefore, several methods have been developed based on two-class SVM to tackle the multiclass problem.

One of the most popular way to solve the multiclass classification problem with K classes is to split the problem into K separate binary classification sub-problems [?]. The i'th sub-problem is defined such that only observations in the i'th class are treated as positive examples, and other observations are treated as negative ones. For example, assume we have multiclass classification problem where the the label vector of training set is defined as

$$\mathcal{Y} = (4, 4, 1, 1, 3, 3, 2, 2).$$

The four corresponding binary classification problem are defined as

$$\begin{aligned}
\mathcal{Y}_1 &= (-1, -1, -1, -1, -1, -1, +1, +1),\\
\mathcal{Y}_2 &= (+1, +1, -1, -1, -1, -1, -1, -1),\\
\mathcal{Y}_3 &= (-1, -1, -1, -1, +1, +1, -1, -1),\\
\mathcal{Y}_4 &= (-1, -1, +1, +1, -1, -1, -1, -1).
\end{aligned}$$

[1] http://www.brenda-enzymes.org/

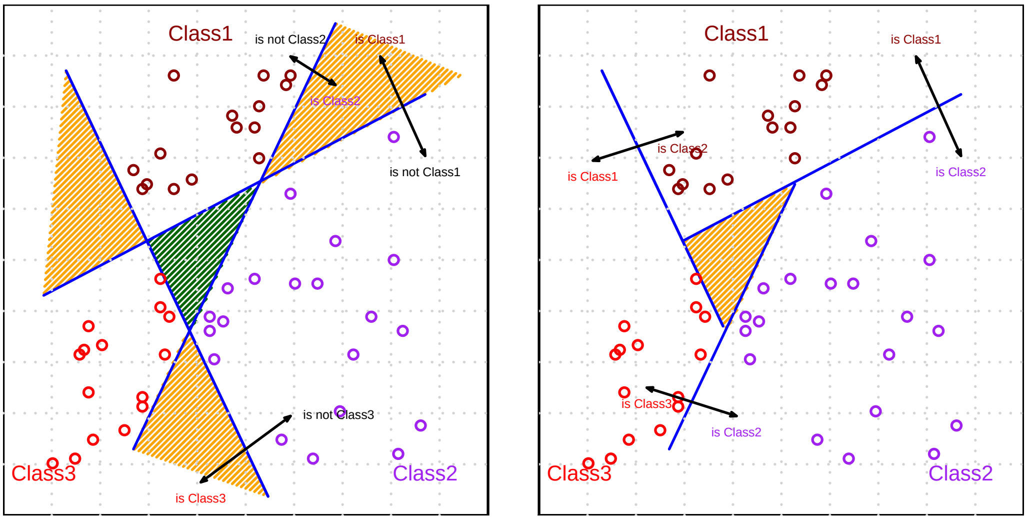


Figure 6: Multiclass support vector machines. Left, points in three classes are separated by one-against-all approach. Inconstancy happens in the areas which belongs to more than one classes, as shown in orange areas. Blue area corresponding to the area resulted by crossing-over of three hyperplanes and therefore belongs to none of the three classes. Right, the inconstancy is addressed by one-again-one approach. Blue area does not exist in this approach. Inconstancy areas become smaller, while still exists.

Once multiclass classification problem is divided, a two-class SVM is trained based on training data of each subproblem. This is known as *one-against-all* approaches.

However, one-against-all approach has consistancy problems when an example is assigned positive target value by several binary classifiers at the same time, shown in the left of Figure 6. The inconsistency can be roughly tackled by taking the class label with the maximum prediction value, defined by

$$(k) = \mathop{\mathbf{argmax}}_{k \in K} f_k(\mathbf{x}).$$

The label assignment schema is still problematic. Since each separate binary classifier is defined based on different optimization problem, the objective values may not be in the same range. There are still examples that cannot be assigned class labels.

Another problem that comes with one-against-all strategy is that the labeling situation is quite biased in each binary classification sub-problem. Assume a multiclass classification problem with equivalent number of examples in five class. A binary classification sub-problem will have 20% of training example be positive labeled and 80% be negative labeled. Therefore, the balanced labeling distribution no more exists.

A more elegant way for multiclass classification is to define the single objective function that considers all K classifiers at the same time [**?**]. The algorithm includes maximizing margin between pair of classes and remaining the number of classes.

However, it needs more time to train such a classifier compared to one-against-all approach.

Another method for multiclass classification is so-called *one-against-one* approach which aims to train $\frac{K(K-1)}{2}$ classifiers between any possible pair of classes [?]. The method assigns a label to an example following a majority vote stager. However, it still suffers from consistancy problem as one-against-all approach when one example is assigned several labels at the same time, shown in right of Figure 6.

Furthermore, there is a method for multiclass classification known as *error-correcting output code* [?]. It can be seen as generalized one-against-one approach. The method trains several binary classifiers, each of which takes several classes as positive class and the complementary classes as negative one. By using a smart way to partition the original classes and combining several binary classifiers, the method can alleviate the inconstancy inherited by multiclass classification problem. It is good alternative to most commonly used multiclass algorithms.

2.7 Multilabel classification

Compared to single label classification which assigns the object to one class when there exist two or more classes (corresponding to binary or multiclass classification), the multilabel classification is to categorize the object simultaneously to several classes [?]. In the last ten years, multilabel classification has become increasingly important in many real world applications, for example music categorization [?], gene and enzyme function annotation [?, ?], image and document classification [?], medical diagnosis [?] and molecular classification.

A variety of models have been introduced for multilabel classification task, including approaches that use a collection of binary classifiers [?, ?, ?]. Other methods use Bayesian framework to combine multiple classifiers based on functional constraints [?, ?]. These methods are usually referred as flat classification approach [?]. However, objects are usually associated with hundred or thousand of labels. Most of the flat classification models do not scale well to such high computational demonds, as they are supposed to learn an individual classier for each possible label. Besides, the dependencies between labels are not explicitly modeled. Multilabels are mostly assumed to be independent in flat classification approaches.

Hierarchical classification emerged in recent years as an elegant way for multilabel classification task [?]. It assumes an hierarchy exists over multilabels, and models the dependencies inside the hierarchy. Especially, hierarchical classification models combined with kernel methods help us manipulate high dimensional feature space without explicitly constructing feature maps. These methods include structured SVM [?], M^{3}N [?], HM^3 [?, ?], output kernel tree [?], and max-margin regression [?].

2.7.1 Max-margin conditional random field (MMCRF)

Max-margin conditional random field (MMCRF) is the method used in this thesis. It is an application of the algorithm developed in [?]. In addition to original algorithm [?], which was restricted to hierarchical tree structure, [?] expands it for general graph structures. The model takes as inputs a Markov network over all labels and the kernel representation of the objects. It uses max-margin optimization for parameter learnings. Belief propagation inference is employed for learning parameters and constructing the predictions.

Mathematically, we assume that training data is comprised by a series of n objects $\mathcal{X} = x_1, x_2, \cdots, x_n$ and a set of labeling vectors $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_k$ which is a Cartesian product of the sets $\mathcal{Y}_j = \{-1, +1\}$ for $j = 1, \ldots, k$. There exists a labeling vector $\mathbf{y}_i = (y_1, y_2, \cdots, y_k) \in \mathcal{Y}$, known as *multilabel*, associated to each object x_i. Each component of multilabel vector $\mathbf{y}_i$ is called a *microlabel*, as denoted by y_j for $j = 1, \ldots, k$. The object and multilabel pair $(x_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y}$ serving as training example is drawn from an unknown joint distribution $P(\mathcal{X}, \mathcal{Y})$. In addition, a pair $(x_i, \mathbf{y})$ where $\mathbf{y}$ is an random labeling vector for object x_i is called *pseudo example*. Training example and pseudo example are not necessarily from the same distribution. The task is to learn a conditional probability $P(\mathbf{y}'|x')$, from where one can infer the most probable labeling vector $\mathbf{y}'$ given an unknown object x'.

As mentioned previously, we have a Markov network $G = (\mathcal{V}, \mathcal{E})$ defined over all labels, where node v_j corresponds to the jth component of multilabel vector $\mathcal{Y}$, and edge $e = (v_j, v_i) \in \mathcal{E}$ denotes the dependency relation between nodes v_i and v_j. The conditional probability $P(\mathbf{y}|x)$ can be modeled in factorization form of the exponential family based on the Markov network, defined as

$$\begin{aligned} P(\mathbf{y}|x) &= \frac{1}{Z(x, \mathbf{w})} \prod_{e \in \mathcal{E}} \exp(\mathbf{w}_e^T \phi_e(x, \mathbf{y}_e)) \\ &= \frac{1}{Z(x, \mathbf{w})} \exp(\mathbf{w}^T \phi(x, \mathbf{y})), \end{aligned}$$

where $\mathbf{w}_e$ is the parameter associated to each edge and $Z(x, \mathbf{w}) = \sum_y \exp(\mathbf{w}^T \phi(x, \mathbf{y}))$ is known as the partition function. $\phi_e(x, \mathbf{y}_e) = \varphi(x) \otimes \varphi_e(\mathbf{y}_e)$ is the tensor product between input feature map $\varphi(x)$ and output feature map $\varphi_e(\mathbf{y}_e)$ which contains all possible labelings on the edge e. The tensor product ensures that no pre-alignment is needed between input and output feature maps. Besides, parameter $\mathbf{w}$ is sensitive to edge labels.

2.7.2 Max-margin learning

Typically in above condition random field (CRF) model, one wants to learn a maximum log likelihood parameter $\mathbf{w}$ defined by

$$\begin{aligned}(\mathbf{w}) &= \underset{\mathbf{w}}{\mathbf{argmax}}\,\mathbf{log}\prod_{i=1}^{n} P(\mathbf{y}_i|x_i,\mathbf{w}) \\ &= \underset{\mathbf{w}}{\mathbf{argmax}}\sum_{i=1}^{n}\big[\mathbf{w}^T\phi(x_i,\mathbf{y}_i) - \mathbf{log}(Z(x_i,\mathbf{w}))\big]. \end{aligned} \tag{19}$$

However, direct optimization of the (19) is usually not an easy task since the partition function $Z(x_i, \mathbf{w})$ is hard to solve. An alternative route is to maximize the ratio of the likelihood of correct training example against one of the most competetive pseudo example defined as

$$\begin{aligned}(\mathbf{w}) &= \underset{\mathbf{w}}{\mathbf{argmax}}\,\mathbf{log}\prod_{i=1}^{n}\frac{P(\mathbf{y}_i|x_i,\mathbf{w})}{\underset{\mathbf{y}\neq\mathbf{y}_i}{\mathbf{max}}P(\mathbf{y}|x_i,\mathbf{w})} \\ &= \underset{\mathbf{w}}{\mathbf{argmax}}\sum_{i=1}^{n}\left[\mathbf{w}^T\phi(x_i,\mathbf{y}_i) - \underset{\mathbf{y}\neq\mathbf{y}_i}{\mathbf{max}}\,\mathbf{w}^T\phi(x_i,\mathbf{y})\right]. \end{aligned} \tag{20}$$

As a result, the maximizing the ratio of log likelihood of training example against one of the most competetive pseudo example is transfered as maximizing the linear margin between them. The maximum odds ratio [**?**] estimation defined in (20) avoids calculating the partition function $Z(x_i, \mathbf{w})$, and is analog to optimization problem of canonical SVM as maximizing linear margin.

In MMCRF, we want the margin here to be scaled according to a loss function, denoted by ℓ_Δ, such that the pseudo examples are push away farther from the training example when more divergences exist between the multilabels. According to hard margin SVM defined by (2), the margin based structured output learning can be stated as the following minimization problem based on hard margin

$$\begin{aligned}(\mathbf{w}) =& \underset{\mathbf{w}}{\mathbf{argmin}}\,\frac{1}{2}||w||^2, \\ &\text{s.t. } \mathbf{w}^T\Delta\phi(x_i,y) \geq \ell_\Delta(\mathbf{y}_i,\mathbf{y}), \\ &\quad 1 \leq i \leq n,\end{aligned}$$

where $\Delta\phi(x_i, y) = \phi(x_i, \mathbf{y}_i) - \phi(x_i, \mathbf{y})$ and $\ell_\Delta(\mathbf{y}_i, \mathbf{y})$ is the hamming loss function indicating the number of nonidentical positions between real labeling vector $\mathbf{y}_i$ and the pseudo labeling vector $\mathbf{y}$. The corresponding soft margin optimization problem is defined as

$$\begin{aligned}(\mathbf{w}) =& \underset{\mathbf{w}}{\mathbf{argmin}}\,\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i, \\ &\text{s.t. } \mathbf{w}^T\Delta\phi(x_i,y) \geq \ell_\Delta(\mathbf{y}_i,\mathbf{y}) - \xi_i, \\ &\quad 1 \leq i \leq n.\end{aligned}$$

The margin slack parameter $\xi = (\xi_1, \xi_2, \cdots, \xi_n)$ tolerate the errors, as described in (9).

The complexity for handling high dimensionality of feature map of input object requires a dual representation defined as

$$\begin{aligned}(\alpha) =& \underset{\alpha>0}{\mathbf{argmax}}\, \alpha^T\gamma - \frac{1}{2}\alpha^T K\alpha, \\ \text{s.t. } & \sum_{\mathbf{y}} \alpha(i, \mathbf{y}) \leq C, \\ & 1 \leq i \leq n,\end{aligned} \tag{21}$$

where $\gamma = \gamma(\mathbf{y}_i, \mathbf{y}), 1 \leq i \leq n$ are the parameters required for each pseudo example $(x_i, \mathbf{y})$, and K is joint kernel map between pseudo examples $(x_i, \mathbf{y})$ taking the form of

$$\begin{aligned}K &= (\Delta\phi(x_i, \mathbf{y}) \cdot \Delta\phi(x_j, \mathbf{y}')) \\ &= ((\varphi(x_i) \otimes \Delta\phi_e(\mathbf{y}_i, \mathbf{y})) \cdot (\varphi(x_j) \otimes \Delta\phi_e(\mathbf{y}_j, \mathbf{y}'))) \\ &= ((\varphi(x_i) \cdot \varphi(x_j)) \cdot (\Delta\phi_e(\mathbf{y}_i, \mathbf{y}) \cdot \Delta\phi_e(\mathbf{y}_j, \mathbf{y}'))),\end{aligned}$$

where $\Delta\phi_e(\mathbf{y}_i, \mathbf{y}) = \varphi_e(\mathbf{y}_i) - \varphi_e(\mathbf{y})$. Hence, K can be obtained from input kernel map in feature space and output kernel map on multilabels.

For each training example $(x_i, \mathbf{y}_i)$, there exists exponential number of pseudo examples $(x_i, \mathbf{y})$. Hence, exponential number of dual variable $\gamma(\mathbf{y}_i, \mathbf{y})$ and constraints $\alpha(x_i, \mathbf{y})$ are required in (21), which makes the direct optimization not feasible.

2.7.3 Marginal dual problem

The exponential number of constraints or dual variables in (21) requires the approach from a different angle where we map the dual variable $\alpha(x_i, \mathbf{y})$ to a set variables defined as the edges of Markov network, known as edge marginals. Given a Markov network $G = (\mathcal{E}, \mathcal{V})$, the marginal of an edge $e \in \mathcal{E}$ is defined as

$$\mu_e(x_i, e, u) = \sum_{\mathbf{y}' \in \mathbf{y}} [\![\varphi(\mathbf{y}') = u]\!] \alpha(x_i, \mathbf{y}'), \tag{22}$$

where $u \in \{--, -+, -+, --\}$ is a possible label of the edge e. The edge marginal variable $\mu_e(x_i, e, u)$ can be seen as the summation over all dual variables $\alpha(x_i, \mathbf{y})$ that have the same labels on edge e. The set of edge marginal variables is called marginal polytope, denoted by $\mathcal{M}$, which is in polynomial dimension with exponential number of vertices. Hence we map a inference problem in exponential dimension into one in polynomial dimension. The corresponding dual representation of (21) is given by

$$(\mu) = \underset{\mu \in \mathcal{M}^n}{\mathbf{argmax}}\, \mu^T \ell - \frac{1}{2}\mu^T K_e \mu, \tag{23}$$

where $\ell = \ell(x_i, e, u)$ is the loss vector associated to each edge marginal variable $\mu(x_i, e, u)$, defined as the number of differences between edge label u and label for edge e in labeling vector $\mathbf{y}$, and K_e is the kernel value between input feature map and the edges.

2.7.4 Optimization and inference

The optimization of (23) takes an iterative form by using conditional gradient algorithm on each object while keeping others fixed. The update direction is given by gradient from current working object. The solution from iterative conditional gradient algorithm is one from the set $\mathcal{M}$ of marginal dual variables. Each marginal dual variable correspond to a pseudo labeling of the Markov network that is most competetive to the real one. Hence, it is equivalent to finding one configuration of the Markov network that maximizes the edge potential, which has approximate solution by loopy belief propagation algorithm (LBP). The approximate solution suffices since the optimization is iterative on all objects. The labeling scheme for a node of the Markov network need to be reconstructed by edge labels, since the inference algorithm is defined on edge labels. Detailed algorithm for optimization and inference can be referred to [**?**].

3 Molecular representations

When using kernel methods on molecules, most work centers on extracting information of atoms, bonds, and their relationships on molecular structures. There exists various representation of molecular structures on different levels, including simple representations in one dimension, graph representation and descriptor representation. In this section, we focus on molecular graph representations and descriptor representations.

3.1 Molecular graph

Labeled graph provides a natural structural representation of molecules, where a node exists for each atom and edges corresponds to chemical bonds. Different atom types (eg. *oxygen*, *carbon*, *nitrogen*, etc) are represented by node labels, and edges are labeled by the corresponding bond types (eg. *single*, *double*, *aromatic*, etc). Mostly, we use undirected graphs instead of directed ones since there are no difference whether a chemical bond goes from one atom to another or vice versa. Graph representations of molecules exist in 2D or 3D space. However, 2D graph is still the most reliable way to represent a molecule. We will focus graph representation in 2D space in the later work. An example of *cocaine* molecule and corresponding graph representation is shown in Figure 7.

Molecular graph can be described in different ways. The most simple way is via SMILES strings[2], which denotes Simplified Molecular Input Line Entry System. It is a specification that uniquely describes the structure of molecules. The theory was mostly developed at DAYLIGHT[3]. SMILES string is a line notation of a molecules without 2D or 3D coordinates, which is suitable for database search. It is supported in most chemical toolboxes that can be identified and transformed to 2D or 3D descriptions of the corresponding molecules. Hydrogen atoms are excluded in SMILES string, other atoms are explicitly represented. Double bonds and triple bonds are denoted by "=" and "#" respectively. For example, the SMILES string of *cocaine* molecule with formula "C9H8O4" is denoted by "`[H][C@]12CC[C@]([H])([C@H]([C@H](C1)OC(=O)c1ccccc1)C(=O)OC)N2C`". In simple words, a SMILES string of a molecule is generated in such a way that the molecule is written as backbone with branches after breaking the cycles in the corresponding 2D graph representation.

The MDL MOL file[4] was developed by Symyx[5]. It is employed in various biochemical databases (eg. KEGG[6], PubChem[7], ect.) as a standard file format to describe the molecules or chemical objects (eg. proteins, enzymes, etc.). A typical MOL file contains header information, connection table (including atoms and bonds in-

[2] http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html

[3] http://www.daylight.com/

[4] http://www.symyx.com/solutions/white_papers/ctfile_formats.jsp

[5] http://www.symyx.com/

[6] http://www.kegg.com/

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

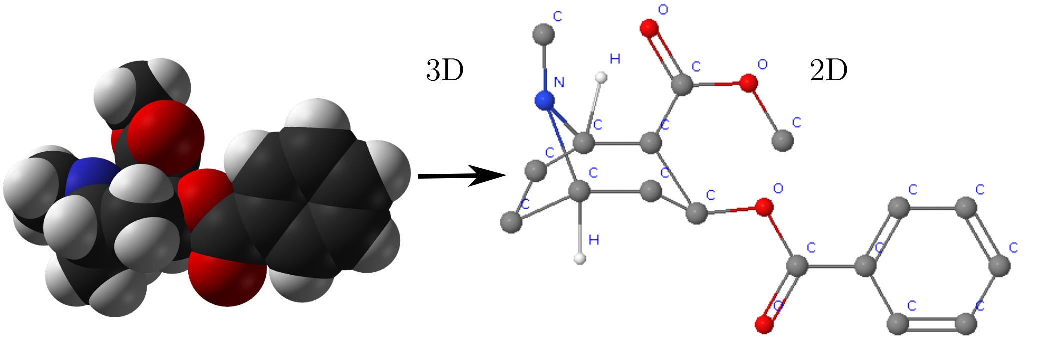


Figure 7: Example of molecular graph representation, structures are drawn by Jmol. Left, molecular structure of *cocaine* is represented as graph in 3D space. Right, the corresponding structure in 2D space.

formation), and the tail section for identifying the end of a MOL file. It is capable of representing the atoms, bonds, as well the connectivity and relationship of a molecule in a detailed manner. The structural data file (SDF) format serves as an extension of MOL file format, and can wrap up MOL files of several molecules. The most important feature of SDF file is that the format allow one to contain various properties of molecules (eg. ID, weight, LogP, etc.)

Another widely used file format is known as MOL2 file[8], originated at TRIPOS[9]. Like MOL file format, MOL2 format also contains atoms, bonds, as well as their connectivity and coordinates of a molecule. Besides, it supports partial charge and isotopes. Furthermore, the MOL2 file describe the atom and bond types in a detailed manner which includes substructure information to increase specificity of the atoms and bonds. An example of MOL and MOL2 file formats is shown in Figure 8.

3.2 Reduced graph

Besides molecular graph, there exists another graph representation of molecules which is known as reduced graph. It is derived from a molecular graph, and initially designed for similarity searching [?]. It is also employed in building structure activity relationship models [?]. It only retains the functional structures of a molecule by performing a series of transformations, for example merging nodes that belong to the same functional units. Two kinds of reduction techniques are frequently used, topological pharmacophore reduction and functional group reduction.

[8] http://www.tripos.com/data/support/mol2.pdf

[9] http://www.tripos.com

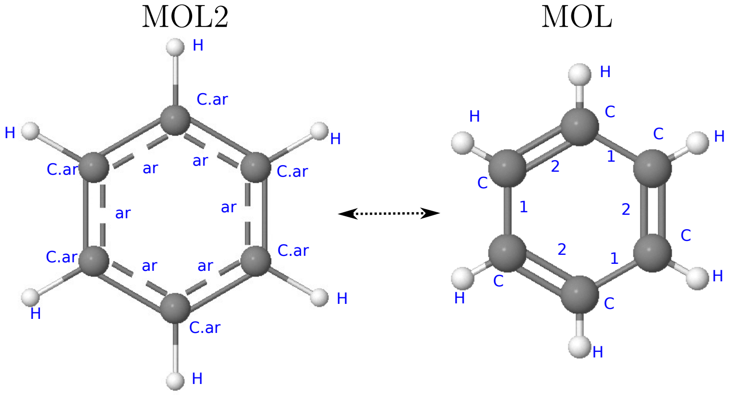


Figure 8: Example of MOL and MOL2 formats of *benzene*. Left, molecular graph in MOL2 format where *aromatic carbon* atoms are explicitly expressed as 'C.ar', and *aromatic* bonds are denoted by 'ar'. Right, corresponding graph in MOL format where only simple atom and bond types are supported.

3.2.1 Topological pharmacophore reduction

Topological pharmacophore reduction technique is to collapse nodes in the molecules that represents the set of feature types. The set of feature types was firstly defined in [?], including ring features and ionization features. Then, it was expanded in [?] by adding up electron state feature (donor or acceptor) and merging joint features, for example the joint donor-acceptor feature. The complete feature set proposed in [?] is shown in Table 1. Figure 9 shows an example of topological pharmacophore reduced graph of *aspirin* molecule, where features are calculate from JoeLib[10] and the priority of features are ordered according to Table 1.

3.2.2 Functional group reduction

Another reduction technique is based on functional groups, which are specific groups of atoms within molecules that are responsible for chemical properties of those molecules. Functional groups are usually defined by domain experts and therefore have different versions with various size. For example PubChem and Open-Babel have definitions for hundreds of functional groups; on the other hand only 49 functional groups are employed in [?]. Derived from topological pharmacophore reduction, functional group reduction is to extract atoms in molecules that belong to functional groups and replace them by special notation. Figure 10 demonstrates the functional group reduction of *aspirin* molecule based on functional group defini-

[10] http://www.ra.cs.uni-tuebingen.de/software/joelib/

Notation	***Ring type***	***Electron state***	***Ionization***
Sc	Aromatic	-	-
Ti	Aromatic	Donor	-
V	Aromatic	Acceptor	-
Cr	Aromatic	Donor&Acceptor	-
Mn	Aromatic	-	Positive
Fe	Aromatic	-	Negative
Co	-	Donor	-
Ni	-	Acceptor	-
Cu	-	Donor&Acceptor	-
Nb	-	-	Positive
Mo	-	-	Negative
Hf	Aliphatic	Donor	-
Ta	Aliphatic	Acceptor	-
W	Aliphatic	Donor&Acceptor	-
Re	Aliphatic	-	Positive
Y	Aliphatic	-	Negative
Zr	Aliphatic	-	-
Zn	-	-	-

Table 1: Topological pharamacophore features.

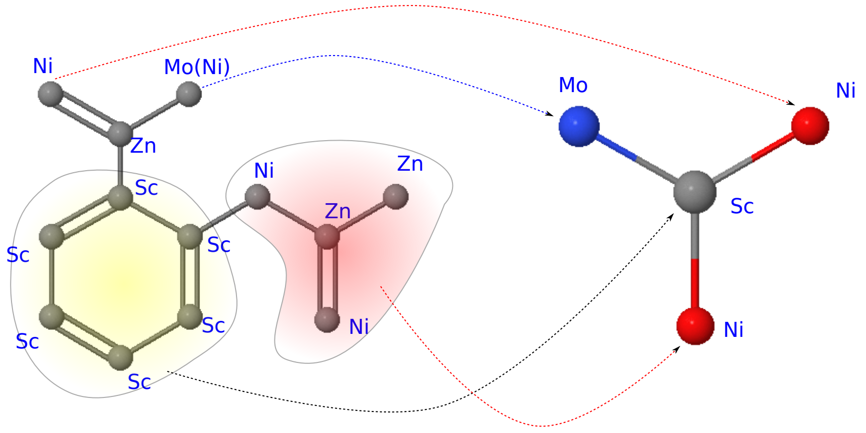


Figure 9: An example of topological pharmacophore reduction. Left, *aspirin* molecule labeled by topological pharmacophore features. Topological pharmacophore features are calculated by JoeLib. Right, corresponding reduced graph with feature type *Zn* excluded.

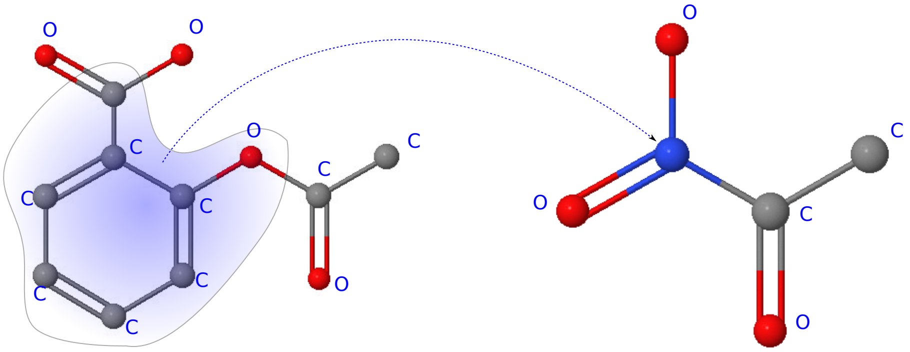


Figure 10: An example of functional group reduction. Left, a functional group with aromatic ring structure is detected in *aspirin* molecule. Right, the corresponding reduced graph with functional group being replaced and other atoms being remained.

tion from PubChem[11]. The priority of functional groups are ordered by size where functional group with more atoms will have higher priority to be extracted.

One problem inherited with reduced graph is the order of pharmacophore features or functional groups to be reduced. Different priority strategy will lead to different reduced graph representation of original molecular graph, especially when one atom can be represented by several features or functional groups. For example, one *oxygen* atom in Figure 9 represents both acceptor and negative ionization features (Mo and Ni). The priority of feature type Mo is higher than Ni. It therefore be replaced by Mo feature type. Currently, there is no good way to solve the problem except arranging the feature types and functional groups by experts.

3.3 Molecular descriptors

Molecular descriptors are designed to encode a molecular structure in a fixed width binary bit vector, which represents the presence or absence of particular substructures or fragments in the molecule. They are extensively used for various tasks in chemical informatics especially for similarity searching, based on the assumption that comparing molecular descriptors will give insight into molecules. There are several variations of molecular descriptors.

3.3.1 Substructure key

One kind of fingerprints that was used first is commonly known as *substructural key*. It is based on pattern matching of a molecular structure to a set of pre-defined substructures. Each substructure becomes a part of the key and has a fixed position in the descriptor space. These substructures are considered to be independent func-

[11] ftp://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubchem_fingerprints.txt

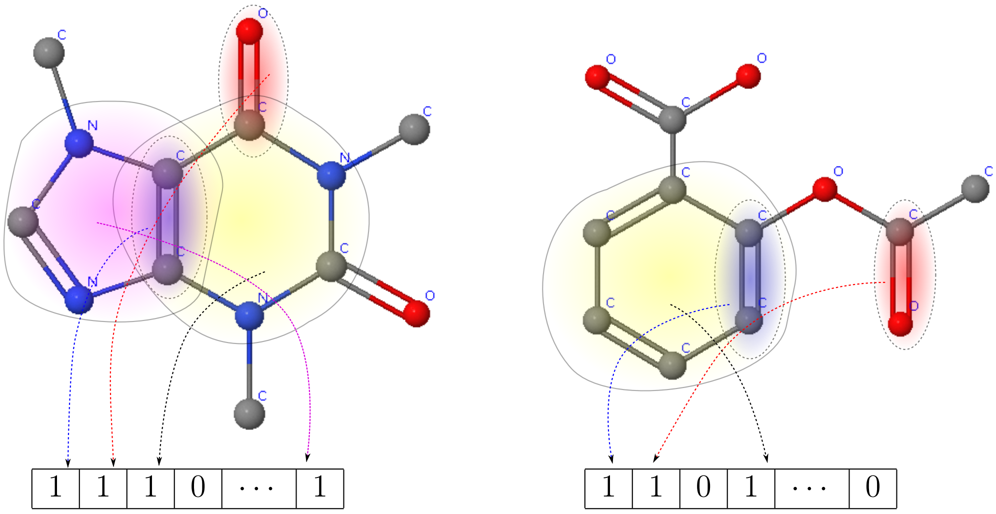


Figure 11: Example of molecular fingerprints. Two molecules are mapped to bit vectors, where positions correspond to substructures.

tional units and are identified based on domain knowledge. Two standards exist for substructure key, MACCS (Molecular Access System) from Symyx and CACTVS [?] from PubChem. Figure 11 illustrates an example of generation and comparison of substructure keys.

However, substructure key suffers from a lack of generality. The substructure set has the critical effect on searching speed as well as the performance of the fingerprints. Substructural keys are usually very sparse since a typical molecule contains just a few substructures.

3.3.2 Hash fragments

An another fingerprint type is called *hash fragments*, which enumerates all cycles or linear fragments up to a certain size in the molecule. The size of the fragments is usually bounded between three to seven. A hash function assigns each of the fragments a hash value, which determines its position in descriptor space. As a result, hash fingerprint is capable of encoding a large number of features in a compact manner. Therefore, it circumvents the sparseness of substructural key. DayLight[12] and OpenBabel[13] offer molecular fingerprints of hash fragment type.

[12] http://www.daylight.com/

[13] http://openbabel.org/wiki/

3.3.3 Pharmacophore fingerprint

Pharmacophore fingerprint is another molecular descriptor that differs from substructure key and hash fragments. It describes the molecules in term of chemical properties other than structural information. In general, it maps molecules to a set of pharmacophore features, each of which has its position in descriptor space like substructure key. The commonly used pharmacophore features include the number of hydrogen bond acceptors (HBA), the number of hydrogen bond donors (HBD), octanol/water partition coefficient (LogP), the number of acidic groups, etc. Molecular descriptors generated from JoeLib[14] are of the pharmacophore fingerprint type.

[14] http://www.ra.cs.uni-tuebingen.de/software/joelib/tutorial/descriptors/descriptors.html

4 Kernels for drug-like molecules

Kernels computed from the structured representation of molecules extend the scope of the traditional approaches by allowing complex derived features to be used (walks, subgraphs, properties) while avoiding excessive computational cost. In this section, we will review several approaches to construct a graph kernel for classification of drug-like molecules, including walk kernel, decomposition kernel, tree kernel, and descriptor kernels.

4.1 Notation

An *undirected graph* $G = (\mathcal{V}, \mathcal{E})$ is comprised by a set of *nodes* $\mathcal{V}$ and set of *edges* $\mathcal{E}$. The size of the node set and the edge set are denoted by $|\mathcal{V}|$ and $|\mathcal{E}|$ correspondingly. Different graphs are distinguished by a subscribe i, denoted by $G_i = (\mathcal{V}_i, \mathcal{E}_i)$. The node set $\mathcal{V}$ contains a sequence of nodes of finite length, denoted by $\mathcal{V} = \{v_1, v_2, \cdots, v_{|\mathcal{V}|}\}$. A edge is defined as $e = (v_i, v_j)$ and edge set is represented by a finite number of edges denoted by $\mathcal{E} = \{e_1, e_2, \cdots, e_{|\mathcal{E}|}\}$. A *walk* of length l in a graph G is denoted by $w = \{v_1, e_1, v_2, e_2, ..., v_l\}$ such that for $i = 1, 2, ..., l-1$ there exists an edge e_i for each pair of nodes (v_i, v_{i+1}).

The $n \times n$ adjacency matrix E of an undirect 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 nodes v_i and v_j (e.g. $(v_i, v_j) \in \mathcal{E}$), and zero otherwise. In an undirected graph, the *neighbor set* of a node v is defined by $\delta(v) = \{u \in \mathcal{V} | (v, u) \in \mathcal{E} \vee (u, v) \in \mathcal{E}\}$. The degree of a node v in an undirected graph is represented by $|\delta(v)|$. Furthermore, $\Delta(G) = \max\{|\delta(v)| : v \in \mathcal{V}\}$ represents the *maximum degree* of an undirected graph G.

4.2 Walk kernel

The first walk kernel was proposed in [?] as marginalized kernel between labeled graphs. The method measures the similarity between a pair of graphs by means of matching random walks up to an infinite length between them. The longer walks are down-scaled by a probability scheme defined on walks.

Walk kernel simulates the trajectory of a random w that starts from one node and keeps jumping from one node to its neighbor. Each walk has a starting probability $P_s(w)$ which is the probability of choosing the starting point of the walk. The starting probability is a uniform distribution over all nodes $\mathcal{V}$ in the graph G. Subsequently in each step, the walk jumps to the neighbor v_j of current node v_i with the transition probability $P_t(v_j|v_i)$, which is also a uniform distribution over all neighboring nodes of current node. It is also possible that a walk does not jump to the neighbors of current node v_i, which requires a stopping probability $P_e(v_i)$. Intuitively, the transition probability $P_t(v_j|v_i)$ and stopping probability $P_e(v_i)$ satisfy

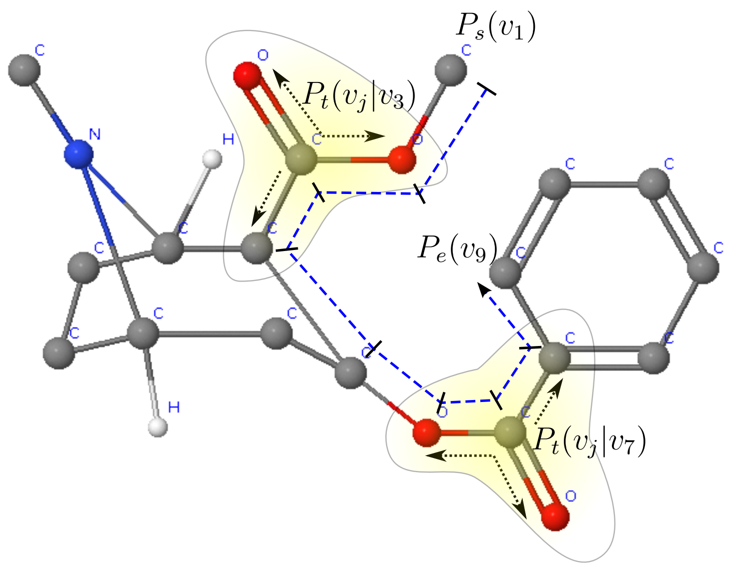


Figure 12: Example of a probability scheme. A random walk of length 9 on *Cocaine* molecule is shown as blue dashed line. The starting probability is denoted by $P_s(v_1)$, and the stopping probability by $P_e(v_9)$. The transition probability of an internal nodes v_i in the walk is defined as uniform distribution over all its neighbors shown as yellow areas. For example, transition probabilities of node v_3 and v_7 are represented by $P_t(v_j|v_3)$ and $P_t(v_j|v_7)$ respectively.

the quantitative relation

$$\sum_{j=1}^{m} P_t(v_j|v_i) + P_e(v_i) = 1,$$

where $|m|$ is the number of the neighboring nodes of the current node v_i. An example of the probability scheme is shown in Figure 12. Given a graph G, the probability of a random walk is therefore defined as

$$P(w|G) = P_s(v_1) \prod_{i=2}^{l} P_t(v_i|v_{i-1}) P_e(v_l),$$

where l is the length of the walk w.

Next, we define the kernel between two random walks w and w' from pair of graph G and G'. Assume w and w' of length l and $l'$are generated according to probability scheme described above. Both of them consists of a sequences of nodes and edges

$$\begin{aligned} w &= (v_1, e_1, v_2, e_2, \cdots, v_{l-1}, e_{l-1}, v_l), \\ w' &= (v'_1, e'_1, v'_2, e'_2, \cdots, v'_{l-1}, e'_{l-1}, v'_l). \end{aligned}$$

The kernel between a pair of random walks is defined as

$$K_w(w, w') = \begin{cases} 0 & \text{if } l \neq l' \\ K_v(v_1, v'_1) \prod_{i=2}^{l} K_e(e_i, e'_i) K_v(v_i, v'_i) & \text{if } l = l' \end{cases},$$

where $K_v(v, v')$ and $K_e(e, e')$ are matching kernels defined on pair of nodes and edges which return 1 if two objects are the same, and 0 otherwise.

The marginalized walk kernel is summing over the the kernels of all possible pairs of walks up to a infinite length between two graphs, defined as

$$K_{mwk}(G_1, G_2) = \sum_{l=1}^{\infty} \sum_{w_1} \sum_{w_2} K_w(w_1, w_2) P(w_1|G_1) P(w_2|G_2), \tag{24}$$

where w_1 and w_2 are random walks of length l in graph G_1 and G_2.

Computing the kernel $K_{mwk}(G_1, G_2)$ by a straight forward enumeration seems to be impossible, since length of walks range from one to infinite. It is shown in [?] that the kernel function defined by (24) has a nested structure. Therefore, one can calculate the kernel function by an iterative fashion which updates current solution until the function converges.

Another approach for calculating marginalized walk kernel employs the concept of product graph and matrix inversions. It was introduced in [?] as an extension of marginalized graph kernel. A direct product graph between two labeled undirected graphs $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2)$ is denoted by $G_\times(G_1, G_2)$. Node set $\mathcal{V}_\times$ and edge set $\mathcal{E}_\times$ of are defined as

$$\begin{aligned}
\mathcal{V}_\times(G_1, G_2) =& \{(v_1, v_2) \in \mathcal{V}_1 \times \mathcal{V}_2, \text{label}(v_1) = \text{label}(v_2)\}, \\
\mathcal{E}_\times(G_1, G_2) =& \{((v_1, v_2), (u_1, u_2)) \in \mathcal{V}_\times \times \mathcal{V}_\times, \\
& \quad (v_1, u_1) \in \mathcal{E}_1 \wedge (v_2, u_2) \in \mathcal{E}_2 \wedge \text{label}(v_1, u_1) = \text{label}(v_2, u_2)\}.
\end{aligned}$$

There is a node in the product graph $G_\times$ for each pair of nodes from G_1 and G_2 with the same label. There is an edge in the product graph $G_\times$ whenever there are edges in G_1 and G_2 connecting the corresponding nodes with the same label. An example of a product graph is shown in Figure 13. As a result, a walk in product graph corresponds to pair of common walks in the parental graphs.

Given a common walk $w_\times = ((v_1, u_2), (v_2, u_2), \cdots, (v_l, u_l))$ of length l in the product graph $G_\times$, the probability scheme can be rephrased based on the product graph including a starting probability $\pi_s((v_1, u_1))$, a transition probability $\pi_t((v_i, u_i)|(v_{i-1}, u_{i_1}))$, and a stopping probability $\pi_e((v_l, u_l))$ defined as

$$\begin{aligned}
\pi_s((v_1, u_1)) &= P_s(v_1) P_s(u_1), \\
\pi_t((v_i, u_i)|(v_{i-1}, u_{i_1})) &= P_t(v_i|v_{i-1}) P_t(u_i|u_{i-1}), \\
\pi_e((v_l, u_l)) &= P_e(v_l) P_e(u_l).
\end{aligned}$$

The corresponding probability matrix is denoted by π_s, π_t and π_e respectively. The probability of a common walk on product graph is defined by:

$$\pi(w) = \pi_s((v_1, u_1)) \prod_{i=1}^{l} \pi_t((v_i, u_i)|(v_{i-1}, u_{i_1})) \pi_e((v_l, u_l)) = \pi_s \prod_{i=1}^{l} \pi_t \pi_e.$$

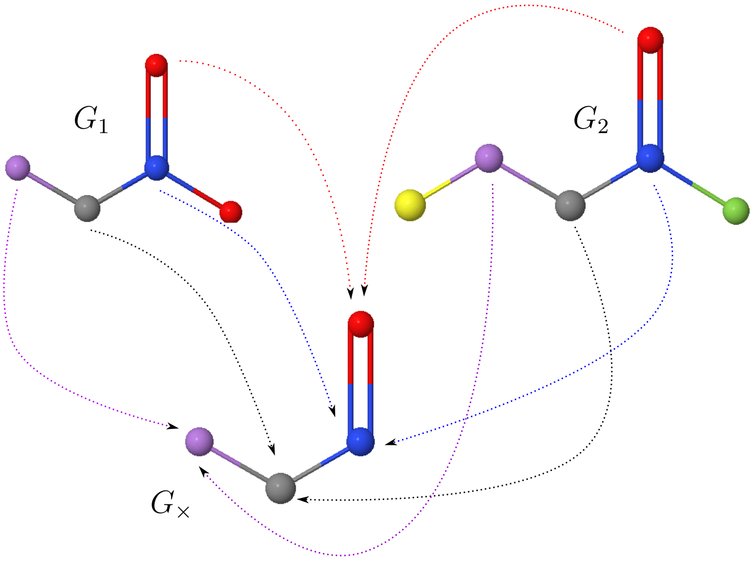


Figure 13: Example of product graph. A product graph $G_\times$ is constructed from parental graphs G_1 and G_2. Nodes of different labels are distinguished by different colors. Edge labels are shown as single or double connections. Nodes in G_1 are matched with ones in G_2. Only matching nodes remained in $G_\times$. Edges between matching nodes existing in both G_1 and G_2 will stay in $G_\times$.

Then, the kernel function is the summation of every possible walk on the product graph which corresponds to all shared random walks on parental graphs, defined as

$$K_{emk}(G_1, G_2) = \sum_{l=1}^{\infty} \sum_{w} \pi(w) = \pi_s^T (I - \pi_t)^{-1} \pi_e^T.$$

Besides marginalized walk kernel, there exists another definition of walk kernel as proposed in [?, ?]. The kernel considers the sum of matching walks in a pair of graphs. The contribution of each matching walk is downscaled exponentially according to its length.

The adjacency matrix has an important property which helps us to calculate the number of walks in graph G up to a finite or infinite length. Note that, if the adjacency matrix E is taken to the power of n, the $(i, j)'$th entry E_{ij}^n shows the number of walks of length n starting from node v_i and ending at v_j.

Walk kernel between pair of graph G_1 and G_2 can be defined based on product graph $G_\times(G_1, G_2)$ and its adjacency matrix $E_\times$ as:

$$K_{wk}(G_1, G_2) = \sum_{i,j=1}^{|\mathcal{V}_\times|} \left[\sum_{i=2}^{\infty} \lambda_i E_\times^n \right]_{i,j}, \tag{25}$$

where $\mathcal{V}_\times$ is the node set of the product graph, $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_l)$ is the sequence of positive downscaling factor that is strictly less than one, and l is the length of walk.

The choice of downscaling factor λ allows the walk kernel function take the form of a geometric or an exponential series expansion. It allows us to solve the matrix power series by matrix inversion operation [**?**] when l is taken to infinity. For example, the limit of a geometric series is:

$$\lim_{l \to \infty} \sum_{i=1}^{l} \lambda^l = \frac{1}{1-\lambda},$$

where $0 < \lambda < 1$. The matrix power series takes the form of geometric series when $\lambda_i = \gamma^i$ $(0 < \gamma < 1)$, and converges if $\gamma < \frac{1}{a}$ where $a \geq \Delta(G_\times)$. In this case, the limit of matrix power series is given by:

$$\lim_{l \to \infty} \sum_{i=1}^{l} \lambda^l E^i = (I - \lambda E)^{-1}.$$

where I is identity matrix with ones on diagonal and zeros elsewhere. The matrix inversion operation has a rough cubic time complexity $O(|\mathcal{V}_\times|^3)$. We notice that the adjacency matrix $E_\times$ is often quite sparse. One can therefore take advantage of methods for sparse matrix to further scale the complexity. However, it is still quite expensive for some molecular graphs. Only several atom types existing in parental graphs makes the size of product graph quite large.

Since longer walks are downscaled by λ_i, the contribution of longer walks are ofter negligible. Figure 14 shows examples of contribution of walks in different length when the matrix power series takes the form of geometric series. Longer walks merely contributes to the final value of the kernel functions. Therefore, we consider finite-length walk kernel where only walks up to length l are explicitly constructed:

$$K_{wk}^{l}(G_1, G_2) = \sum_{i,j=1}^{|\mathcal{V}_\times|} \left[\sum_{i=2}^{l} \lambda_i E_\times^n \right]_{i,j}.$$

Equivalently, it can be defined in a dynamic programming fashion by:

$$K_{dwk}^{l}(G_1, G_2) = \sum_{i=1}^{l} \sum_{v_{\times i} \in V_\times} D_i(v_{\times i}), \tag{26}$$

where $D_i(v_i)$ is calculated by:

$$\begin{aligned} D_0(v_i) &= 1, \\ D_i(v_i) &= \sum_{(v_{\times i}, v_{\times j}) \in E_\times} D_{i-1}(v_{\times j}). \end{aligned}$$

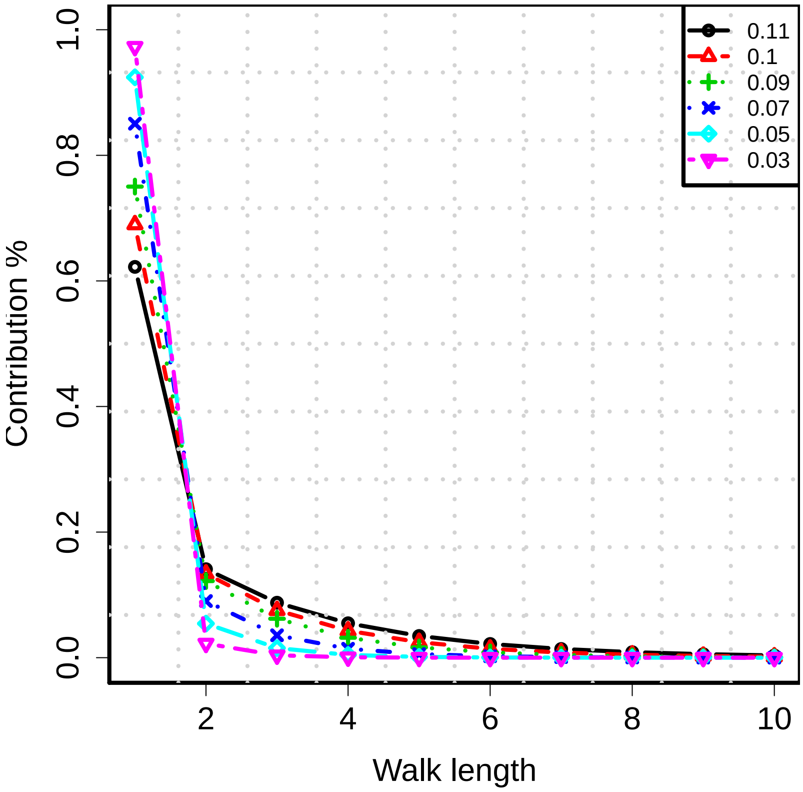


Figure 14: Contribution of walks. Here, we considered the walk kernel of two molecules. Molecular structures and the product graph are not shown. The maximum in-degree or out-degree of product graph in this case is 11. Therefore, the kernel function taking a geometric series form converges when $0 < \lambda < \frac{1}{11}$. Walk kernel based on walk of finite length (up to 50) was calculated. The contribution of walks in different length (up to 10) are explicitly shown as percentage with λ ranging from 0.03 to 0.11. Longer walks, for example walks of length larger than 8, do not contribution much to kernel function and therefore can be ignored during the calculation.

4.3 Weighted decomposition kernel

Weighted decomposition kernel is an instantiation of convolution kernel [?] on molecular graphs. It was introduced in [?] and successfully applied in classification of drug-like molecules in [?]. Like other convolution kernels, weighted decomposition kernel also relies on the decomposition of a molecular graph into a set of subgraphs and the composition of the kernels from a pair of subgraphs. The kernel between a pair of subgraphs are weighted by means of a exact matching kernel.

Specifically, weighted decomposition kernel is based on a set of decompositions of graph G denoted by $D(s, c, G)$. Selector s corresponds to the atom in graph G. c is called contextor, which is the subgraph of G consisting of the surrounding atoms of selector s. It is usually defined with parameter r, known as contextor radius, as the set of atom that can be reached by selector s by a path with length r. An example of graph decomposition is shown in Figure 15.

A weighted decomposition kernel sums over kernels from each pair of contextor defined as

$$K_{wdk}(G_1, G_2) = \sum_{\substack{v_i, c_i \in D^{-1}(G_1) \\ v_j, c_j \in D^{-1}(G_2)}} \delta(v_i, v_j) K_{sub}(c_i, c_j), \tag{27}$$

where $D^{-1}(G)$ is inverse function of decomposition operation $D(s_i, c_i, G)$ that returns all tuples (s_i, c_i) satisfying $(s_i, c_i) \in D(s_i, c_i, G)$. $\delta(x_i, x_j)$ is matching kernel defined on pair of selectors and can takes various forms, for example

$$\delta(v_i, v_j) = \begin{cases} 1 & \text{if } v_i = v_j \\ 0 & \text{if } v_i \neq v_j \end{cases} .$$

$K_{sub}(c_i, c_j)$ is the subgraph kernel that measures the similarity between pair of contextors c_i and c_j. The subgraph kernel discards the structural information and focuses the coincident of the attributes in pair of contextors. It is the summary of kernels from individual attribute and can be defined in different form as:

$$K_{sub}(c_i, c_j) = \sum_{i=1}^{n} K^i_{attri}(c_i, c_j),$$

$$K_{sub}(c_i, c_j) = \prod_{i=1}^{n} K^i_{attri}(c_i, c_j),$$

$$K_{sub}(c_i, c_j) = \sum_{i=1}^{n} (1 + K^i_{attri}(c_i, c_j)),$$

where n is the number of attributes (eg. atom type, bond type, atom charge, etc.) in the contextor. $K^i_{attri}(c_i, c_j)$ is the kernel function for the ith individual attribute and a variation of histogram interaction kernel. It counts the number of common values in each attribute, defined as

$$K^i_{attri}(c_i, c_j) = \sum_{j=1}^{m_i} \min(p_i(j), p'_i(j)),$$

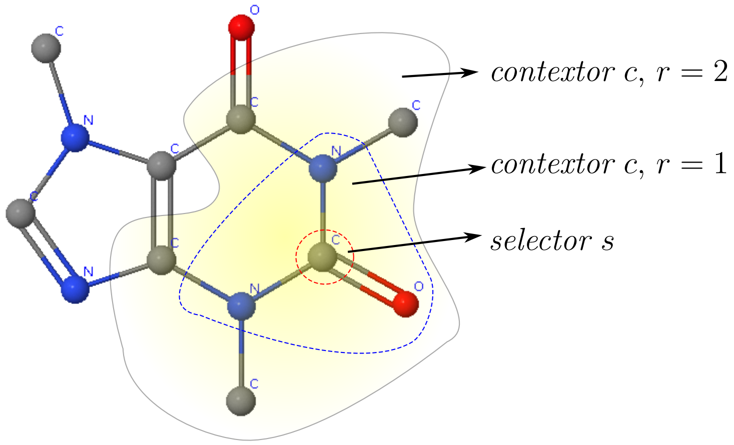


Figure 15: Example of graph decomposition. Molecular structure of *Caffeine* is shown in figure, where gray atom denotes *carbon*, red atom denotes *Oxygen* and blue atom denotes *nitrogen*. Single or double bonds are also illustrated. A decomposition centers on a *carbon* atom shown as selector. The radius $r = 1$ contextor and radius $r = 2$ contextor are the areas inside the blue curve and the black curve.

where m_i is the number of possible values for attribute i, and $p_i(j)$ is the occurrence of value j for attribute i. An example of histogram intersection operation related to weighted decomposition kernel is shown in Figure 16.

4.4 Tanimoto kernel

Once the molecules are represented as fingerprints, one still need to define the similarity between fingerprints. The similarity measurement should be independent of whether either are hash fingerprints, substructure keys or others. Several kernel functions that measure the similarity of molecular fingerprints were proposed in [?] including: *Tanimoto kernel, MinMax kernel and Hybrid kernel.*

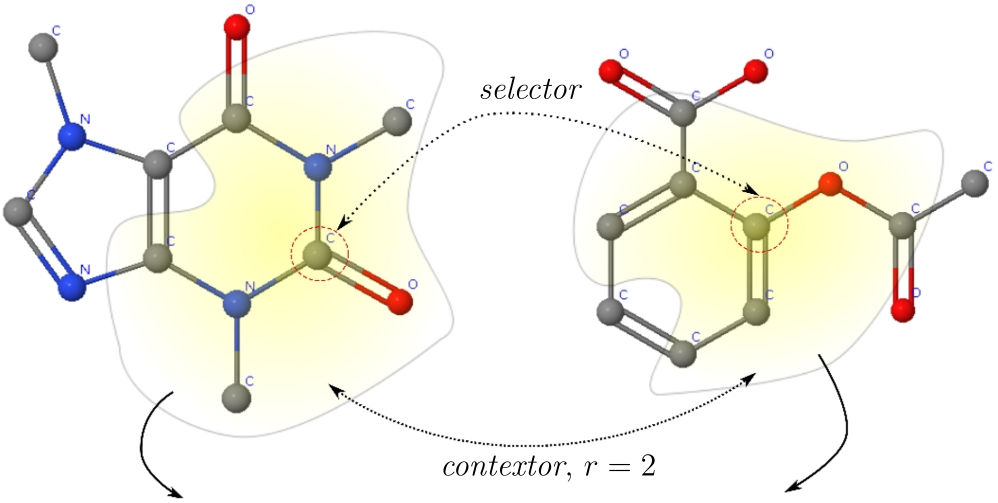
Tanimoto kernel, which is derived from Tanimoto coefficient, is specially tailored for measuring the similarity between two binary bit vectors. It is the most popular one among the three and is employed in various chemical toolboxes (eg. OpenBabel[15], DayLight[16], JoeLib[17], etc.).

Given two molecules u and v, as well as the corresponding fingerprints in form of the binary bit vectors $\phi(u)$ and $\phi(v)$ in the same dimension, Tanimoto kernel is defined

[15] http://openbabel.org/wiki/

[16] http://www.daylight.com/

[17] http://www.ra.cs.uni-tuebingen.de/software/joelib/



Carbon	selector
Carbon	5
Nitrogen	2
Oxygen	1

Carbon	selector
Carbon	7
Nitrogen	0
Oxygen	1

Figure 16: Example of histogram intersection operation. Left is the molecular structure of *Caffeine* with one of its radius $r = 2$ decomposition. Right is the molecular structure of *Aspirin* with one of its $r = 2$ decomposition. The histogram of contextors of both molecule is shown in the table. The intersection kernel sums over all three atom types taking the minimum value of both histogram, which is $5+0+1 = 6$.

as

$$K_{TK}(u,v) = \frac{(\phi(u) \cdot \phi(v))}{(\phi(u) \cdot \phi(u)) + (\phi(v) \cdot \phi(v)) - (\phi(u) \cdot \phi(v))}, \tag{28}$$

where $(\phi(u) \cdot \phi(v))$ denotes the inner product and can be interpreted as the shared 1-bits on two binary vectors $\phi(u)$ and $\phi(v)$. Intuitively, Tanimoto kernel simply measures the ratio between the number of elements in the intersection of two vector $\phi(u)$ and $\phi(v)$ and the number of elements in the union of the two vector $\phi(u)$ and $\phi(v)$. The kernel function returns a value between 0 and 1, where 1 indicates pair of vectors $\phi(u)$ and $\phi(v)$ are identical and 0 means they are opposite.

4.5 MinMax kernel

Tanimoto kernel is suitable for the cases where molecules are mapped to binary bit vector which corresponds to the present or absent of functional groups, fragments, paths or cycles in the molecules. However, pair of molecules with same binary fingerprints may still vary greatly, for example two molecules with same functional groups but with different numbers. Therefore, using counts other than binary values seems to be more reliable for measuring the similarity between molecules, especially for molecules with different size. The counts in the fingerprint vector indicate the actual number of the objects (eg. paths, cycles, etc.) that exists in the molecules. MinMax kernel [**?**] is designed to meet the need of introducing counts in molecular fingerprints.

Given two molecules u and v, and the corresponding fingerprints $\tilde{\phi}(u)$ and $\tilde{\phi}(v)$ with counts in the same dimension, MinMax kernel is defined as

$$K_{MK}(u,v) = \frac{\sum_{pos \in \mathcal{P}} \mathbf{min}(\tilde{\phi}(u), \tilde{\phi}(v))}{\sum_{pos \in \mathcal{P}} \mathbf{max}(\tilde{\phi}(u), \tilde{\phi}(v))},$$

where **max** and **min** operations result in the vectors that take the maximum or minimum value of each position from two input vector $\tilde{\phi}(u)$ and $\tilde{\phi}(v)$. $\mathcal{P}$ denotes the all positions in fingerprint vector $\tilde{\phi}(u)$.

MinMax kernel works exactly the same as Tanimoto kernel on the molecular fingerprints in form of binary bit vectors. Fingerprints with counts can be translated to an extended fingerprints in form of binary bit vectors by replacing the count in each position with a number of position according to the count. Given two fingerprints $\tilde{\phi}(u)$ and $\tilde{\phi}(v)$ with counts, the transformation t and the corresponding extended fingerprints are defined as

$$\tilde{\phi}(u) = (x_{u1}, x_{u2}, \cdots, x_{un}) \xrightarrow{t} \phi(u) = (\overbrace{y_{11}, y_{12}, \cdots}^{max(x_{u1},x_{v1})}, \overbrace{y_{21}, y_{22}, \cdots}^{max(x_{u2},x_{v2})}, \cdots, \overbrace{y_{n1}, y_{n2}, \cdots}^{max(x_{un},x_{vn})}), \tag{29}$$

where $x_i \in \mathbb{R}$ and $y_i \in \{0, 1\}$. Therefore, MinMax kernel on fingerprints with counts can be interpreted as Tanimoto kernel on the corresponding extended fingerprints, shown in Figure 17.

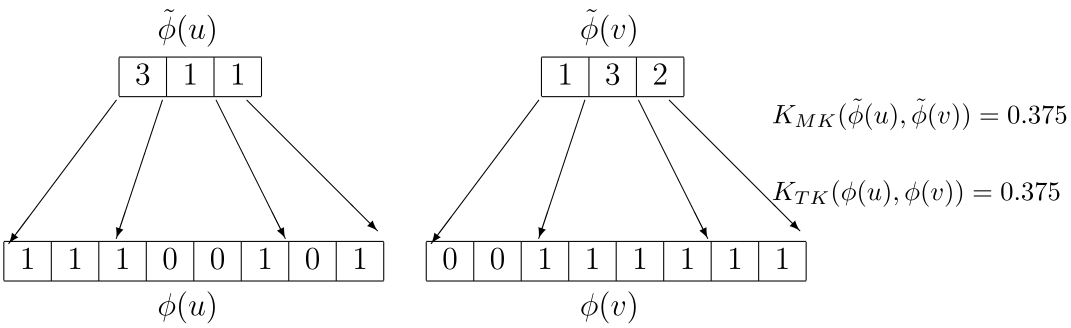


Figure 17: MinMax kernel example. Given two molecules **u** and **v** together with their fingerprints with counts $\tilde{\phi}(u)$ and $\tilde{\phi}(v)$, the value from MinMax kernel function is 0.375. Transform the fingerprints with counts into ones in form of binary bit vector according to (29). The similarity between two molecules can be measured equivalently by using Tanimoto kernel over transformed fingerprints $\phi(u)$ and $\phi(v)$.

4.6 Hybrid kernel

Tanimoto kernel focuses on the shared objects in pair of molecules. MinMax kernel considers also the number of shared objects. However, neither of them includes the information about the objects that are absent in both of the molecules. Hybrid kernel [?] combines two Tanimoto kernels to measure the number of objects presents in pair of molecules as well as the number of objects that are missing in both of them. Given two molecule **u** and **v** as well as the corresponding fingerprints in form of binary bit vector $\phi(u)$ and $\phi(v)$ in the same dimension, Hybrid kernel is defined as

$$K_{HK}(u,v) = \frac{1}{3}((2-\theta)\cdot K_{TK}(u,v) + (1+\theta)\cdot \neg K_{TK}(u,v)),$$

where $\neg$ is logical negation of binary vector and $\theta \in [-1,+2]$. Hybrid kernel turns to Tanimoto kernel when $\theta = -1$. It should be clear that Hybrid kernel is still valid when replacing Tanimoto kernel with MinMax kernel. It, therefore, works also on the fingerprints with counts.

4.7 Tree kernel

Tree kernel was first introduced in [?], which focus on subtree patterns extracted from graphs. We consider a kernel between two molecular graphs $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2)$. Assume $v_1 \in \mathcal{V}_1$ and $v_2 \in \mathcal{V}_2$, subtrees of height h rooted at v_1 and v_2 are denoted by $T^h_{v_1}$ and $T^h_{v_2}$. A neighbor matching set $\mathcal{M}(v_1,v_2)$ of pair of root nodes in two subtrees $T^h_{v_1}$ and $T^h_{v_2}$ is defined as

$$\begin{aligned}\mathcal{M}'(v_1,v_2) = \{&R \subseteq \zeta(v_1)\times\zeta(v_2), R \neq \emptyset \\ &\wedge (\forall (v_a,v_b),(v_c,v_d) \in R, v_a = v_c \Leftrightarrow v_b = v_d) \\ &\wedge (\forall (v_a,v_b) \in R), \text{label}(v_a) = l\text{abel}(v_b) \wedge \text{label}(e_{1a}) = \text{label}(e_{2b})\},\end{aligned}$$

where $\zeta(v)$ represents the set of neighbors of node v. Each $R \in \mathcal{M}'$ contains identical neighbors of v_1 and v_2, and corresponds to the tree pattern rooted at v_1 and v_2 of height h in graph G_1 and G_2.

The subtree matching kernel between pair of subtrees $T_{v_1}^h$ and $T_{v_2}^h$ of height h is defined recursively as:

$$K_t^1(T_{v_1}^1, T_{v_2}^1) = \begin{cases} 1 & \text{if } v_i = v_j \\ 0 & \text{if } v_i \neq v_j \end{cases},$$

$$K_t^h(T_{v_1}^h, T_{v_2}^h) = \lambda_{v_1}^h \lambda_{v_2}^h \sum_{R \in \mathcal{M}(v_1, v_2)} \prod_{(v_1', v_2') \in R} K_t^{h-1}(T_{v_1'}^{h-1}, T_{v_2'}^{h-1}),$$

where λ_v is the positive down-scaling factor strictly less than 1. The value λ is related to the height of the subtree to ensure that the contributions from bigger trees are balanced with ones from smaller trees.

Tree kernel between two graph G_1 and G_2 is defined as

$$K_{tree}(G_1, G_2) = \lim_{h \to \infty} \sum_{v_1 \in V_1} \sum_{v_2 \in V_2} K_t^h(T_{v_1}, T_{v_2}).$$

The kernel sums over all pair of subtrees rooted from each pair of nodes in two graphs.

5 Experimental data

In this section, we will introduce the datasets we employ in our experiments. For NCI-cancer datasets, we will describe the sampling technique we used to construct different version of datasets. We will also describe the auxiliary data, mainly microarray profiling data of cancer cell lines, which we used to construct the Markov network over the cell lines.

5.1 Mutag dataset

Mutag dataset is based on data from a review on the literature of mutagenicities in *Salmonella Typhimurium* based on 230 aromatic and heteroaromatic nitro compounds [?]. The determinants for mutagenecity in the study were hydrophobicity and energies of the lowest unoccupied molecular orbitals. As a result, 188 congeners were extracted together with their structure-activity relationship (SAR) data. Therefore, Mutag dataset is very suitable for machine learning and is widely used as one of the standard datasets.

Specifically, 125 molecules in Mutag dataset are labeled by +1 as positive and 63 are labeled by −1 as negative. Distribution of molecules in two classes according to the number of atoms and chemical bonds are shown in Figure 18 and Figure 19. Mutag dataset is publicly available at ChemDB[18].

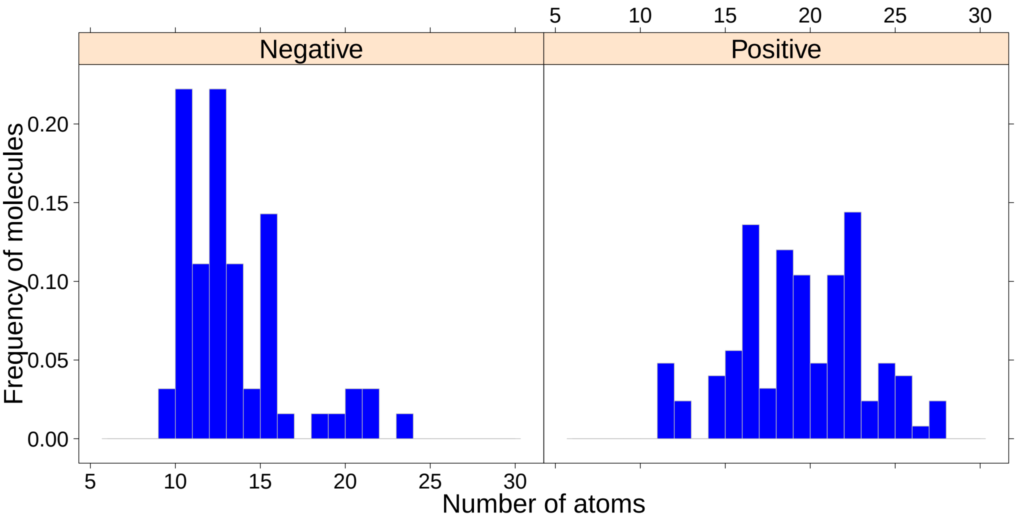


Figure 18: Distribution molecules according to the number of atoms in Mutag dataset. Left, atom distribution of molecules in negative class. Right, atom distribution of molecules in positive class.

[18] ftp://ftp.ics.uci.edu/pub/baldig/learning/mutag/

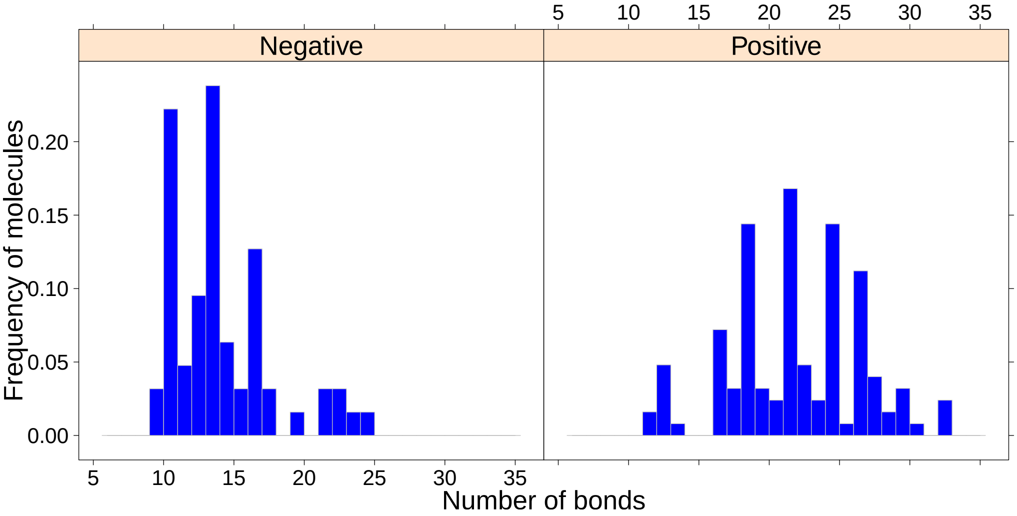


Figure 19: Distribution of molecules according to the number of bonds in the Mutag dataset. Left, bond distribution of molecules in the negative class. Right, bond distribution of molecules in the positive class.

5.2 NCI-cancer datasets

Developmental Therapeutics Program[19] (DTP) from National Cancer Institute and National Institutes of Health[20] (NCI/NIH) was designed to screen up to 3,000 compounds every year searching for potential anti-cancer drugs. This program utilizes bioactivity information of large number of molecules against several human cancer cell lines including leukemia, melanoma and cancers of the lung, colon, brain, ovary, breast, prostate, and kidney. Molecular structure and activity data can be obtained through PubChem Bioassay database[21] [?]. For each molecule tested against a certain cell line, the dataset provides a bioactivity outcome that we use as the classes (active, inactive).

NCI-cancer datasets were first employed in early study [?] of predicting mutagenecity, toxicity and anti-cancer activity by independently sampling for each subset approximately equivalent number of active molecules and inactive ones from original datasets. The datasets resulted from sampling were mostly used in the subsequent researches. However, we found the sampled datasets were erroneous and lots of molecules were mislabeled compared to the latest version of PubChem Bioassay database. Therefore, we discard the old datasets and reconstruct the datasets that was employed in the following experiments. The reconstruction is directly based on PubChem Bioassay database.

Currently, there are 43,884 molecules in the PubChem Bioassay database together with anti-cancer activities in 73 cell lines. 59 cell lines have screening experimental

[19] http://dtp.nci.nih.gov/
[20] http://www.cancer.gov/
[21] http://www.ncbi.nlm.nih.gov/pcassay

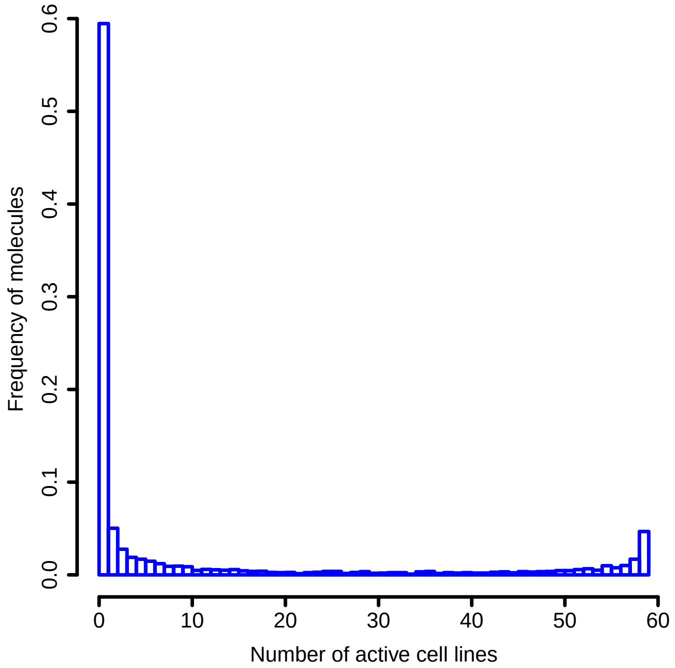


Figure 20: Skewness of the multilabel distribution. Molecules are sorted according to the number of cell lines they are active against, from zero to 59. Label on the x-axis denotes the number of cell lines that molecules are active against.

results for most molecules and $4,554$ molecules have no missing data in these cell lines, therefore these cell lines and molecules are selected and employed in our experiments. The number of active molecules as well as inactive ones in each subset of NCI-cancer datasets are reported in Table 2.

However, molecular activity data are highly biased over the cell lines. Figure 20 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.

Figure 21 shows a heatmap of a normalized Tanimoto kernel, where molecules have been sorted by the number of cell lines they are active against. The heatmap shows that the molecules in the two extremes of the multilabel distribution form groups of high similarity whereas the molecules in the middle are much more dissimilar both to each other and to the extreme groups. The result seems to indicate that the majority of molecules in the dataset are either very specific or very general in the targets they are active against. Other kernels mentioned in section 4 produce a similar heatmap indicating that the phenomenon is not kernel-specific.

Cell line	***Active***	***Inactive***	***Cell line***	***Active***	***Inactive***
NCI-H23	831	3723	NCI-H226	711	3843
NCI-H322M	647	3907	NCI-H460	953	3601
HOP-62	760	3794	HOP-92	743	3811
NCI-H522	1104	3450	A549/ATCC	765	3789
EKVX	599	3955	LOX-IMVI	1005	3549
M14	863	3691	MALME-3M	823	3731
UACC-62	861	3693	UACC-257	677	3877
SK-MEL-2	678	3876	SK-MEL-5	903	3651
SK-MEL-28	621	3933	PC-3	783	3771
DU-145	733	3821	SF-268	804	3750
SF-295	806	3748	SF-539	852	3702
SNB-19	661	3893	SNB-75	760	3794
U251	866	3688	HT29	857	3697
COLO205	863	3691	HCT-15	859	3695
KM12	808	3746	HCC-2998	765	3789
HCT-116	1002	3552	SW-620	979	3575
MCF7	1027	3527	MDA-MB-435	899	3655
MDA-N	888	3666	BT-549	670	3884
T-47D	759	3795	NCI/ADR-RES	706	3848
MDA-MB-231	681	3873	HS-578T	685	3869
OVCAR-3	859	3695	IGROV1	801	3753
SK-OV-3	634	3920	OVCAR-4	631	3923
OVCAR-5	558	3996	OVCAR-8	858	3696
RPMI-8226	1074	3480	SR	1357	3197
CCRF-CEM	1377	3177	K-562	1144	3410
MOLT-4	1207	3347	HL-60(TB)	1321	3233
A498	663	3891	CAKI-1	827	3727
RXF393	836	3718	786-0	908	3646
ACHN	848	3706	TK-10	578	3976
UO-31	766	3788			

Table 2: Number of active and inactive molecules in NCI-cancer datasets with all molecules.

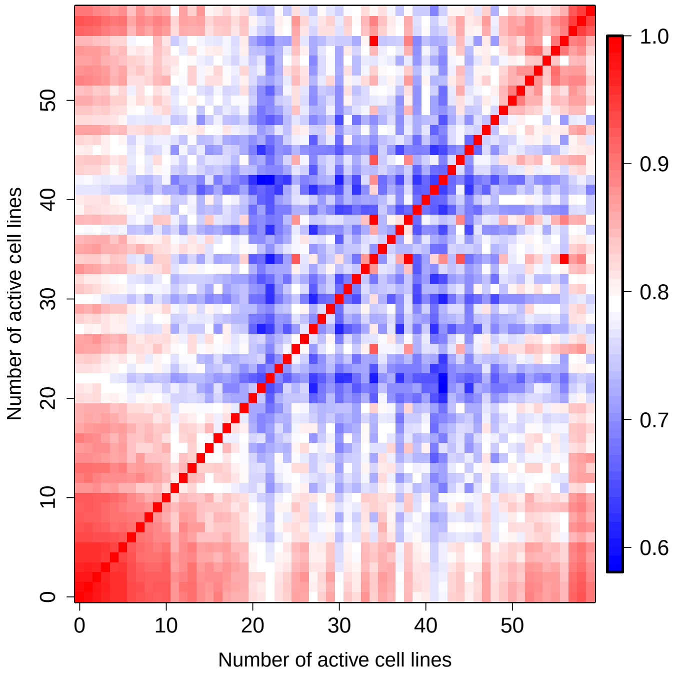


Figure 21: Heatmap of the kernel space for the molecules sorted by the multilabel distribution. Label on axises denotes the number of cell lines that molecules are active against.

Because of the above-mentioned skewness, we prepared different versions of the dataset:

Full. This dataset contains all 4, 554 molecules from the NCI-cancer dataset with their activity class (active vs. inactive) recorded against all 59 cancer cell lines.

No-Zero-Active. In this dataset, we removed all molecules that are not active towards any of the cell lines (corresponding to the leftmost peak in Figure 20). The remaining 2, 305 molecules are all active against at least one cell line. The number of the molecules that are active or inactive against each subsets of the dataset is shown in Table 3.

Middle-Active. In order to circumvent the skewness and concentrate on the most interesting molecule, we also followed the preprocessing procedure suggested in [**?**], and selected the molecules that are active against more than 10 cell lines and inactive against more than 10 cell lines. Middle-active means the molecules that are active against average number of cell lines. As a result, 544 molecules remained and were employed in our experiments. The number of active and inactive molecules in dataset of this version is shown in Table 4.

Cell line	***Active***	***Inactive***	***Cell line***	***Active***	***Inactive***
NCI-H23	831	1474	NCI-H226	711	1594
NCI-H322M	647	1658	NCI-H460	953	1352
HOP-62	760	1545	HOP-92	743	1562
NCI-H522	1104	1201	A549/ATCC	765	1540
EKVX	599	1706	LOX-IMVI	1005	1300
M14	863	1442	MALME-3M	823	1482
UACC-62	861	1444	UACC-257	677	1628
SK-MEL-2	678	1627	SK-MEL-5	903	1402
SK-MEL-28	621	1684	PC-3	783	1522
DU-145	733	1572	SF-268	804	1501
SF-295	806	1499	SF-539	852	1453
SNB-19	661	1644	SNB-75	760	1545
U251	866	1439	HT29	857	1448
COLO205	863	1442	HCT-15	859	1446
KM12	808	1497	HCC-2998	765	1540
HCT-116	1002	1303	SW-620	979	1326
MCF7	1027	1278	MDA-MB-435	899	1406
MDA-N	888	1417	BT-549	670	1635
T-47D	759	1546	NCI/ADR-RES	706	1599
MDA-MB-231	681	1624	HS-578T	685	1620
OVCAR-3	859	1446	IGROV1	801	1504
SK-OV-3	634	1671	OVCAR-4	631	1674
OVCAR-5	558	1747	OVCAR-8	858	1447
RPMI-8226	1074	1231	SR	1357	948
CCRF-CEM	1377	928	K-562	1144	1161
MOLT-4	1207	1098	HL-60(TB)	1321	984
A498	663	1642	CAKI-1	827	1478
RXF393	836	1469	786-0	908	1397
ACHN	848	1457	TK-10	578	1727
UO-31	766	1539			

Table 3: Number of active and inactive molecules in NCI-cancer datasets with zero-active molecules excluded.

Cell line	***Active***	***Inactive***	***Cell line***	***Active***	***Inactive***
NCI-H23	272	272	NCI-H226	158	386
NCI-H322M	129	415	NCI-H460	336	208
HOP-62	222	322	HOP-92	197	347
NCI-H522	380	164	A549/ATCC	217	327
EKVX	109	435	LOX-IMVI	381	163
M14	299	245	MALME-3M	246	298
UACC-62	295	249	UACC-257	154	390
SK-MEL-2	154	390	SK-MEL-5	302	242
SK-MEL-28	132	412	PC-3	228	316
DU-145	196	348	SF-268	256	288
SF-295	255	289	SF-539	289	255
SNB-19	153	391	SNB-75	194	350
U251	301	243	HT29	293	251
COLO205	279	265	HCT-15	319	225
KM12	230	314	HCC-2998	202	342
HCT-116	408	136	SW-620	396	148
MCF7	395	149	MDA-MB-435	276	268
MDA-N	275	269	BT-549	165	379
T-47D	224	320	NCI/ADR-RES	224	320
MDA-MB-231	163	381	HS-578T	178	366
OVCAR-3	284	260	IGROV1	226	318
SK-OV-3	124	420	OVCAR-4	151	393
OVCAR-5	79	465	OVCAR-8	308	236
RPMI-8226	353	191	SR	460	84
CCRF-CEM	446	98	K-562	407	137
MOLT-4	418	126	HL-60(TB)	430	114
A498	140	404	CAKI-1	255	289
RXF393	257	287	786-0	329	215
ACHN	291	253	TK-10	114	430
UO-31	221	323			

Table 4: Number of active and inactive molecules in NCI-cancer datasets only with middle-active molecules.

5.3 Auxiliary data and Markov network

In order to use MMCRF to classify drug molecules we need to build a Markov network for the cell lines used as the output, with nodes corresponding to cell lines and edges denoting potential statistical dependencies. To build such a network, we employed high throughput data of various kinds on NCI-cancer cell lines. NCI-cancer cell lines are widely studied and comprehensively profiled at DNA, RNA, protein, mutation, functional, and pharmacological levels [?, ?]. Multiple profiling datasets were integrated in [?] and made publicly available from NCI database[22].

Pearson correlation coefficient is a measurement of the correlation between two variables. Given two random variables a and b with their expectations μ_a and μ_b and standard deviations σ_a and σ_b, Pearson correlation ρ_{ab} is defined as

$$\rho_{ab} = \frac{E[(a-\mu_a)(b-\mu_b)]}{\sigma_a \sigma_b},$$

where E is the expectation operation. Pearson correlation is good at revealing linear relationship and used in [?] as a measurement of dependencies between two microarray profilings. Here, we constructed a correlation matrix between pairs of cell lines and extract the Markov network from the matrix by favouring the high value pairs. We considered the following three methods for network extraction:

Maximum weight spanning tree. Take the minimum number of edges that make a connected network whilst maximizing the edge weights. Spanning trees based on several microarray profiling data are shown in Figure 22.

Correlation thresholding. Take all edges that exceed a fixed threshold. This approach typically generates a general non-tree graph. Figure 23 depicts the Markov networks over cell lines generated by correlation thresholding methods based on several microarray profiling data.

Graphical lasso estimation. Graphical lasso (Glasso) is an algorithm that is able to estimate a sparse graph from a inverse covariance matrix [?]. However, we leave this direction for future exploration.

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

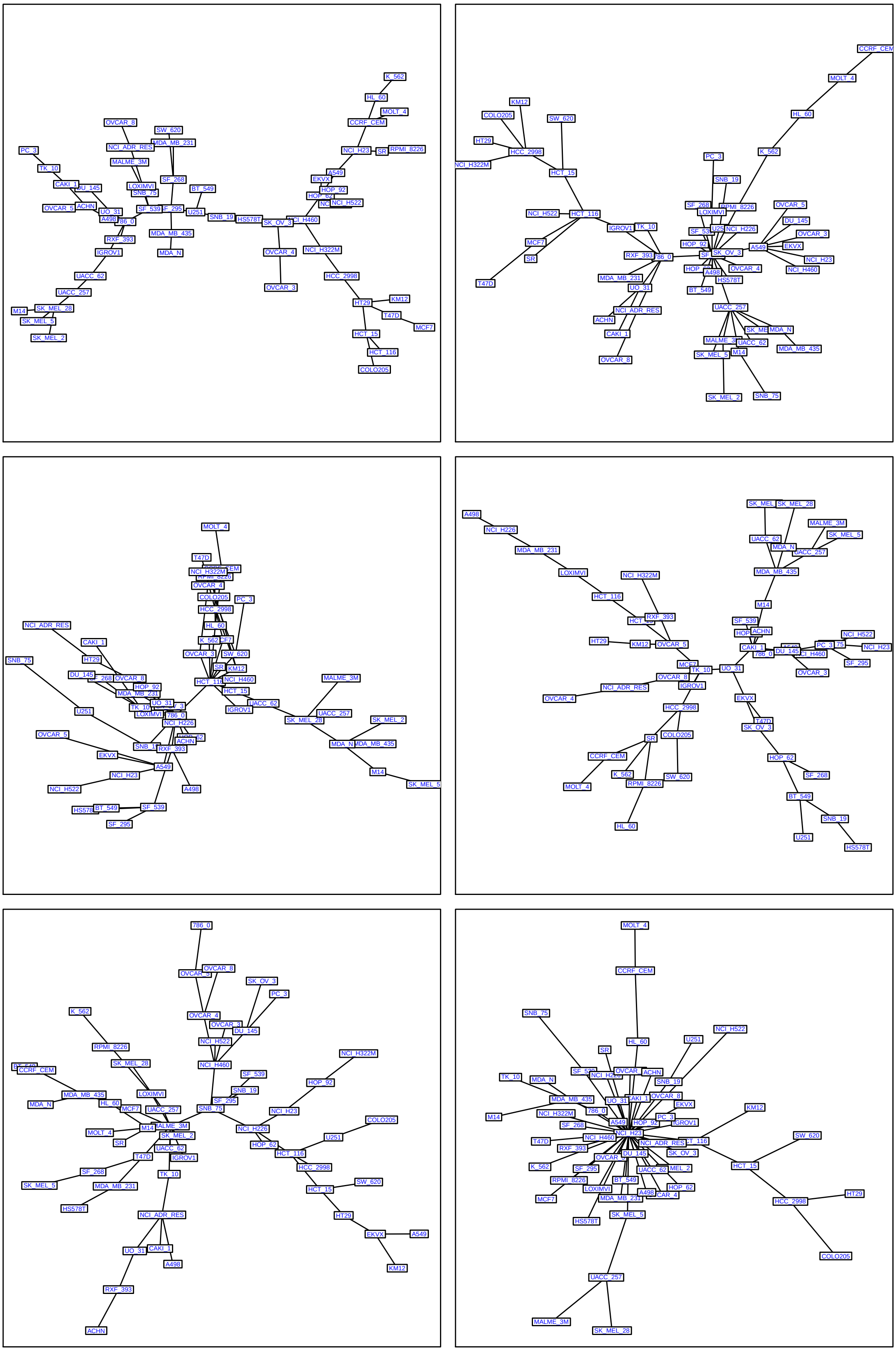


Figure 22: From left to right, top to bottom are spanning tree over cell lines based on reverse-phase lysate arrays, cDNA arrays, Affymetric HU6800 arrays, miRNA arrays, ABC transporter arrays and Affymetric U133 arrays [?].

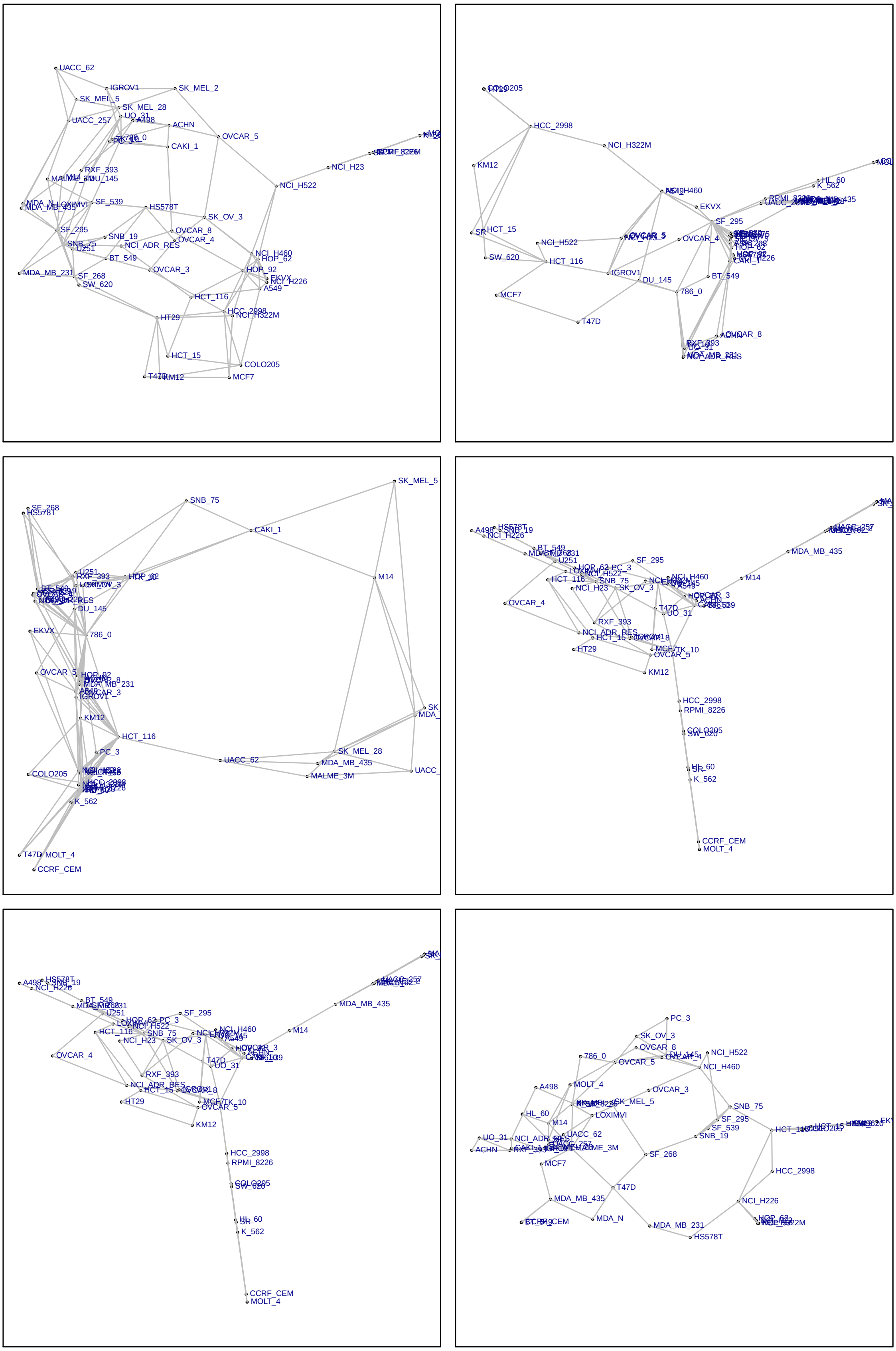


Figure 23: From left to right, top to bottom are Markov network over cell lines by correlation thresholding based on reverse-phase lysate arrays, cDNA arrays, Affymetric HU6800 arrays, miRNA arrays, ABC transporter arrays and Affymetric U133 arrays [?].

6 Experimental results

We implemented several kernel methods, described in Section 4, for classification purposes of drug-like molecules. We also built up binary and structural classifiers with MMCRF and SVM algorithms, and tested their performances on several datasets. However, there are numbers of parameters that need to be adjusted before we can actually carry out our experiments. We also need to define the measurements for comparing different classifiers.

In this section, we will first introduce the standards we used in our experiments for evaluation of different classifiers. Then, the choices of kernel parameters and SVM parameters will be briefly discussed. After that, we will focus on binary classification problem in order to find the best classifier on the problem. Finally, we will devote to structural classification problem and demonstrate the improvements we got compared to binary classification.

6.1 Measurement of success

6.1.1 Statistical measurement

In order to achieve fair comparisons over different classifiers, we need to define the standards we will use as the measurements of the performances. For a binary classification problem, where data are labeled either *positive* or *negative*, there are four possible outcomes. If a positive example is labeled also as positive by the classifier, it is called *true positive* (tp). If the predictor gives a negative label to an example which is actually positive, it is then known as *false negative* (fn). Conversely, it is called *true negative* (tn) if an actual negative example being labeled as negative. *False positive* (fp) means a negative example is labeled as positive. The corresponding confusion matrix is defined in Table 5. Here, we consider several different statistical measurements to evaluate the behaviors of classifiers:

Accuracy. The prediction accuracy (ACC) is the ratio of the number of correct predictions (true positive and true negative) against the whole population, defined as

$$ACC = \frac{tp + tn}{tp + fp + tn + fn}.$$

Accuracy is suitable for the balanced datasets which have almost equivalent number of positive and negative examples. When datasets are not balanced, the predictions are dominated by ones from bigger classes which leads to a high accuracy but bad classifier performance. Accuracy is bounded between zero and one, where one means all predictions are correct.

Precision and recall. Precision (PRE) is the ratio of true positive predictions

		Real value		
		Positive	Negative	
Prediction	Positive	*true positive*	*false positive*	*sensitivity*
	Negative	*false negative*	*true negative*	*specificity*

Table 5: Prediction confusion matrix.

against all positive predictions, defined as

$$PRE = \frac{tp}{tp + fp}.$$

A precision $PRE = 1$ means all positive prediction are correct (nothing about negative predictions). On the other hand, recall (REC) is the ratio of true positive predictions against all real positive examples, defined as

$$REC = \frac{tp}{tp + fn}.$$

F1 score. F1 score (F_1) is the weighted average of precision and recall. The commonly used F1 score is defined as

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC},$$

where precision and recall are evenly weighted.

True positive rate and true negative rate. True positive rate (TPR) or *sensitivity* is the proportion of true positive predictions against all positive examples, which is the same as recall. Similarly, true negative rate (TNR) or *specificity* is the proportion of negative prediction against all negative examples, defined as

$$TNR = \frac{tn}{tn + fp}.$$

ROC curve and AUC score. A receiver operating characteristic (ROC) curve is the plot of true positive rate against true negative rate. AUC score (AUC) is the area below ROC curve, which is equivalent to the probability that the classifier give a higher score to a randomly picked positive example than to a negative one. AUC score is most often used in machine learning since it also tolerances the unbalanced datasets. A random classifier will have $AUC = 0.5$, and a perfect classifier corresponds to $AUC = 1$.

Two tailed sign test. In statistics, the two tailed sign test can be used to test the hypothesis that there is no significant differences for two random variables, when we randomly draw pair of examples from them. The hypothesis will be rejected if the value of the statistics is either sufficiently small or sufficiently large.

6.1.2 Validation methods

In machine learning, the statistical measurements are usually reported from a *k-fold cross validation* procedure. In a k-fold cross validation, the dataset is randomly divided into k subsets. Each subset is served as validation dataset for testing the classifier, and the rest $k-1$ subsets are served as training data for learning a classifier at the same time. This is repeated k times, until each subset is used exactly once as a training set. The cross validation procedure ensure there is no bias generated during learning phrase.

The *leave-one-out cross validation* is a special version of k-fold cross validation. It uses each single example as testing data and the rest of examples as training data, until all examples in the dataset are tested once. It is a validation method that commonly used for small datasets. Therefore the statistical measurements on Mutag dataset are reported from a leave-one-out cross validation procedure.

NCI-cancer dataset is high biased in the sense that molecules are not well distributed according to the number of cell lines they are active against, as shown in Figure 20. Therefore, we adopted a stratified 5-fold cross validation procedure on NCI-cancer dataset. We arrange molecules into 60 groups according to the number of cell lines they are active against, and divide molecules of each group into 5 subsets. In each iteration, we use one subset from each group as validation data and the rest of the subsets as training data. It is then repeated five times, until all data are taken once as testing examples. Therefore, we further address the skewness of the cancer dataset.

6.2 Experimental setup

We used graph representation for molecules in our experiments. Hydrogen atoms are excluded in our graph representations. MOL2 files are also employed to improve the performances of the classifiers. We did not use reduced graph representation. However, it is still quite an interesting approach, and we will explore it in the future work.

Kernel parameters corresponds to the set of features we put into consideration. Therefore, they are very important in our experiments. We focus on walk kernel with walks of finite length, and calculate it in a dynamic programming fashion. Since optimized parameters for finite length walk kernel was not mentioned in any publications, we explicitly test the parameters of walk kernel on Mutag dataset, and keep the optimized parameter in later experiments. The maximum length of the walk is decided by parameter l, with the range of $(1 \leq l \leq 11)$ in our experiments. The down scaling parameter λ of walk kernel takes the value from 0.01 to 0.11 with a step of 0.01. For the weighted decomposition kernel, we use the version with neighbors of contextor radius three and atom type property only, as recommended in [?]. Hash fragment features are extracted explicitly for all fragments of length 6, which is the default value in Openbabel. Functional group features also follow the

Notation	***Kernel method***
WK	Walk kernel with finite length of walks
WDK	Weighted decomposition kernel
HF	Hash fragment features
SK	Substructure key features
TK	Tanimoto kernel

Table 6: Notations for different kernel methods.

functional group definition in OpenBabel. No additional parameters were needed for Tanimoto kernel. All kernels are normalized. For the following experiments, we use notations for different kernel methods other than their full names. Notations are described in Table 6.

We used the SVM implementation of the libSVM software package written in C++[23]. We tested various values of C ranging from one to 100. Relative hard margin ($C = 100$) emerged as the value we used in subsequent experiments. We used $\xi = 0.01$ in our experiments.

6.3 Binary classification

6.3.1 Mutag dataset

With the basic experimental designs, we first tried to optimize walk kernel parameters l and λ on Mutag dataset. In the first experiment, we constructed walk kernel on Mutag dataset with fixed $l = 10$ and various λ parameters that took values from 0.01 to 0.11 with a step of 0.01. SVM parameter C was fixed on $C = 100$. We report in Figure 24 the accuracies, AUC scores and F1 scores of different λ from a leave-one-out cross validation. The optimized λ parameter for walk kernel is 0.11, which can be seen from the results.

Then, we fix $\lambda = 0.11$, and tested the effect of l parameter. We took different l parameters ranging from 1 to 10. The accuracies, AUC scores and F1 scores are reported in Figure 25. As demonstrated in the results, different l lead to almost the same performance, while $l = 5$ gives the best results. Therefore, the parameters for walk kernel were set to $\lambda = 0.11$ and $l = 5$, which achieve the accuracy of 81.38%, AUC score of 87.05% and F1 score of 85.71%. The classification accuracy and AUC score match the state-of-the-art performance on Mutag dataset described in [**?**].

Next, we considered other kernel methods mentioned in Table 6. We explicitly constructed these kernels on Mutag dataset, where walk kernel took the parameter settings from the previous experiments and parameters of other kernels followed the

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

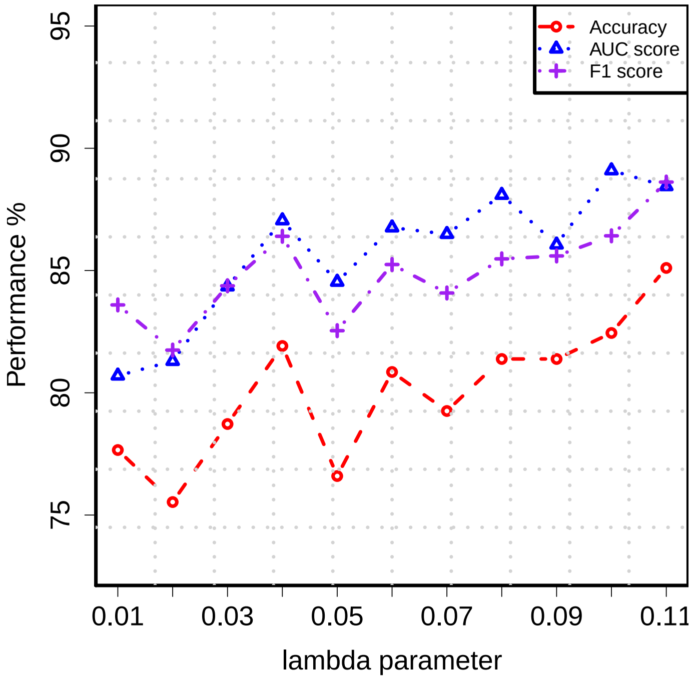


Figure 24: Parameter λ of walk kernel on Mutag dataset.

Classifier	***Accuracy***	***AUC score***	***F1 score***
TK+HF	**86.17%**	**89.94%**	**89.76%**
TK+FG	84.57%	86.57%	88.63%
WDK	81.38%	89.18%	86.06%
WK	81.38%	87.05%	85.71%

Table 7: Performances of classifiers on Mutag dataset.

description in Section 6.2. We report classification accuracies, AUC scores and F1 scores in Table 7.

Overall, the best kernel methods that works on Mutag dataset is Tanimoto kernel (TK) with hash fragments features (HF), which achieves the best accuracy, AUC score and F1 score in our experiments. Tanimoto kernel with functional group features behaves similarly to the one with hash fragment features. While, walk kernel and weighted decomposition kernel are almost equivalent and are not as good as the Tanimoto kernel.

6.3.2 NCI-cancer dataset

We continued to test the behaviors of different kernel methods on binary classification of NCI-cancer dataset. We have prepared three versions of NCI-cancer dataset,

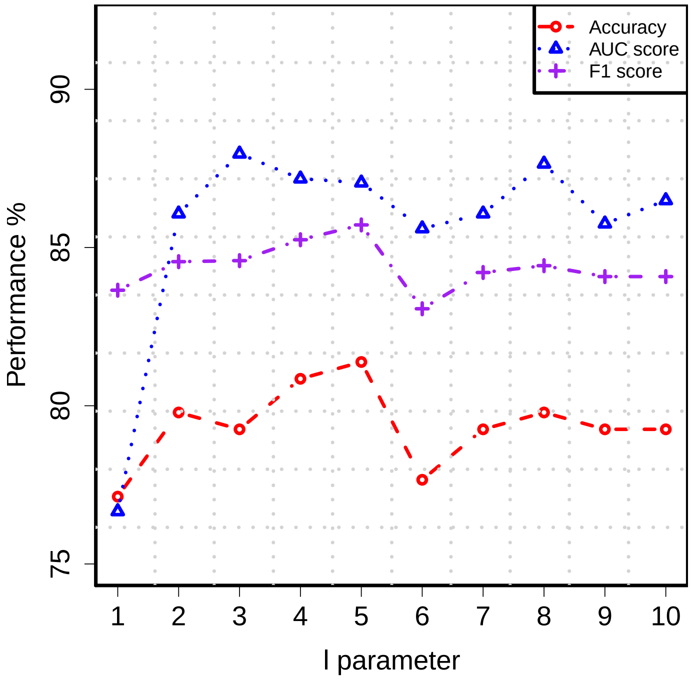


Figure 25: Parameter l of walk kernel on Mutag dataset. λ is set to 0.11.

Classifier	***Accuracy***	***PRE***	***REC***	***F1 score***
TK+HF	64.13%	57.45%	50.25%	52.74%
TK+FG	62.55%	54.52%	**54.31%**	**54.06%**
WDK	63.86%	56.83%	49.73%	51.64%
WK	**64.31%**	**58.00%**	46.20%	47.67%

Table 8: Performance of classifiers on NCI-cancer dataset with middle-active molecules.

as proposed in Section 5.2. Kernels were explicitly constructed, where walk kernel took the optimized parameter from previous section and parameters of other kernels followed the setting illustrated in Section 6.2.

In the first experiment, we used the version of NCI-cancer dataset which includes only the middle-active molecules. We report average accuracy, precision, recall and the F1 score of cell lines from different classifiers in Table 8[24]. Tanimoto kernel (TK) with hash fragment features (HF) and functional group features (FG) achieve the competitive results and slightly outperform walk kernel (WK) and weighted decomposition kernel (WDK).

[24] NA values are generated in precisions and corresponding F1 scores of some cell lines from WDK classifiers. The average precision and F1 scores of WDK are calculated after removing all NA values.

Classifier	*Accuracy*	*PRE*	*REC*	*F1 score*
TK+HF	**72.00%**	**68.86%**	**60.71%**	**64.86%**
TK+FG	69.61%	66.45%	58.20%	61.99%
WDK	61.05%	55.37%	36.19%	42.91%
WK	62.23%	57.38%	39.38%	45.86%

Table 9: Performance of classifiers on NCI-cancer dataset with no-zero-active molecules.

Classifier	*Accuracy*	*PRE*	*REC*	*F1 score*
TK+HF	**80.34%**	**73.41%**	**51.99%**	**60.73%**
TK+FG	79.63%	72.19%	49.32%	58.48%
WDK	69.46%	36.88%	7.10%	11.57%
WK	70.12%	43.85%	8.67%	13.99%

Table 10: Performance of classifiers on NCI-cancer dataset with full data.

In the second experiment, we used NCI-cancer dataset without zero-active molecules. Table 9 shows the average accuracies, precisions, recalls and F1 scores from the experiments, where Tanimoto kernel with hash fragment features outperformed other kernel methods. In the following experiments, we uses the dataset with all molecular data which results in similar results, as shown in Table 10.

Overall from our experiments on binary classification problem, Tanimoto kernel with hash fragment features outperformed other kernel methods in almost all statistical measurements over all versions of our datasets. It should be noticed that SVM C parameter was set with $C = 100$ in all of our experiments in order to achieve fair comparisons of various kernel methods. Increasing C, which corresponds to expanding the margin, sometimes improve the performance of the classifiers.

It is shown that applying polynomial kernel or Gaussian kernel over current kernels will also improve their F1 scores [**?**]. Polynomial kernel was also employed as a standard setting for kernel methods in [**?**, **?**]. However, we did not use any polynomial or Gaussian kernel recomposition since we only wanted to compare different kernel methods.

6.4 Multilabel structural classification

Compared to binary classification which predicts a single label at a time, multilabel structural classification manages to predict all labels in one shot. The classifier also includes information on dependencies among single labels represented as the structure. In this section, we will focus on multilabel structural classification, and make the comparisons against binary classifiers constructed by SVM. The NCI-cancer dataset, which is a multilabeled dataset, will be employed in the experiments. Since Tanimoto kernel with hash fragments features was proved to be the best kernel from binary classification experiments, we will also used it to construct MMCRF classifiers.

6.4.1 Effect of Markov network

We report in Figure 26 the prediction accuracies on dataset with middle-active molecules using various Markov networks, as describe in Section 5.3. The prediction accuracies on different Markov network structures differ slightly. The best accuracy was given by using maximum weighted spanning tree approach on RNA radiation array dataset, shown in Figure 27, which describes profiles of radiation response in cell lines. This meets our expectations since cancer cells mostly mutated from normal cells and normal cells with radiation treatments can possibly explain the mutations.

We also used random graphs over cell lines. However, predictions based on random Markov networks were not able to give as good results as the ones constructed from auxiliary data.

6.4.2 Effect of dataset version

We compare the performances of SVM and MMCRF classifiers in term of prediction accuracies and F1 scores on three versions of NCI-cancer dataset. We show overall accuracy and microlabel F1 score of MMCRF versus SVM for each cell line in Figure 28. Points above the diagonal line correspond to improvements in accuracies or F1 scores by MMCRF classifier. MMCRF improves the F1 score over SVM on each version of the data in statistically significant manner, as judged by the two-tailed sign test shown in Table 11. Accuracy is improved in two versions, No-Zero-Actives and the Middle-Active molecules, again in a statistically significant manner. Among the Middle-Active dataset, the difference in accuracy (bottom, left of Figure 28) is sometimes drastic, around 10 percentage units in favor of MMCRF for a significant fraction of the cell lines.

We categorize molecules into different groups according to their activities against the number of cell lines. We compare the accuracy and F1 score of MMCRF and SVM in each activity group on different versions of the dataset. The results[25] are shown in Figure 29. In the first two versions of the dataset where biased molecular data exist, MMCRF classifier is always better than SVM classifier when the density of labels is high, which corresponds to the molecules that are active against many cell lines. They both made competitive predictions when the label density is low. While for the last version of dataset where biased molecules are removed, MMCRF

[25] Average accuracies and F1 scores are calculated from each interval, shown as x-axis, in order to alleviate the random fluctuation.

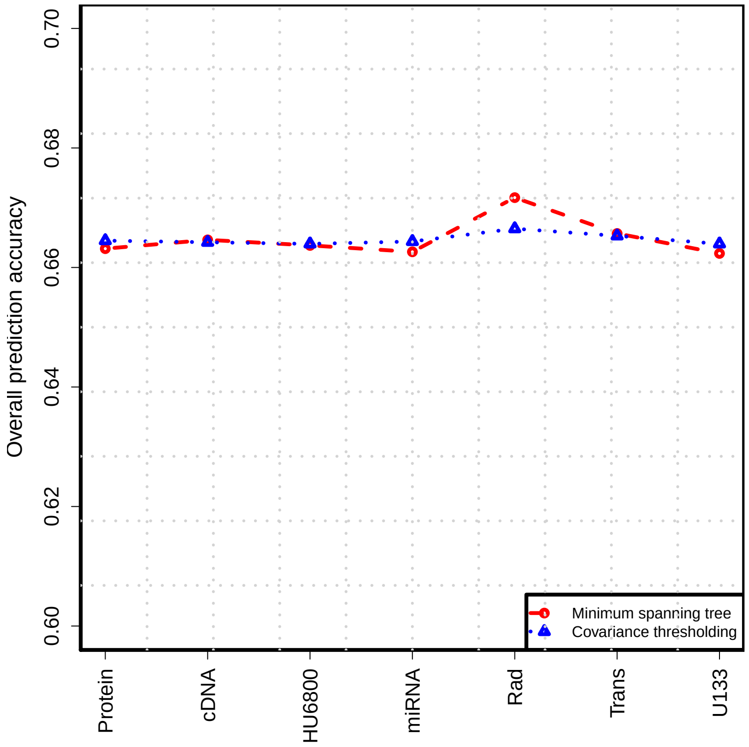


Figure 26: Effects of Markov network construction methods and type of auxiliary data (from left to right: reverse-phase lysate arrays, cDNA arrays, Affymetric HU6800 arrays, miRNA arrays, RNA radiation arrays, ABC transporter arrays, and Affymetrix U133 arrays).

classifier clearly outperforms SVM in all cases. The results further demonstrate that the predictions from SVM are highly affected by the biased molecules, the ones active against no cell lines or all cell lines.

6.4.3 Agreements of predictions from SVM and MMCRF

In molecular classification, positive predictions are more interesting than negative ones since they correspond to potential drug candidates. Therefore, we further check the positive and negative predictions from MMCRF and SVM on three versions of the datasets. Table 12 shows the results on dataset with full molecular data, Table 13 depicts the results by excluding zero-active molecules, and Table 14 illustrates the results by using only middle-active molecules. The results are calculated from the stratified 5-fold cross validation. Overall, MMCRF and SVM classifiers agree on most of positive and negative predictions (approximately 90% and 95% respectively). MMCRF is better at predicting positive labels, while SVM is better at negative ones. Since the original dataset is highly biased with 90% of molecules being inactive

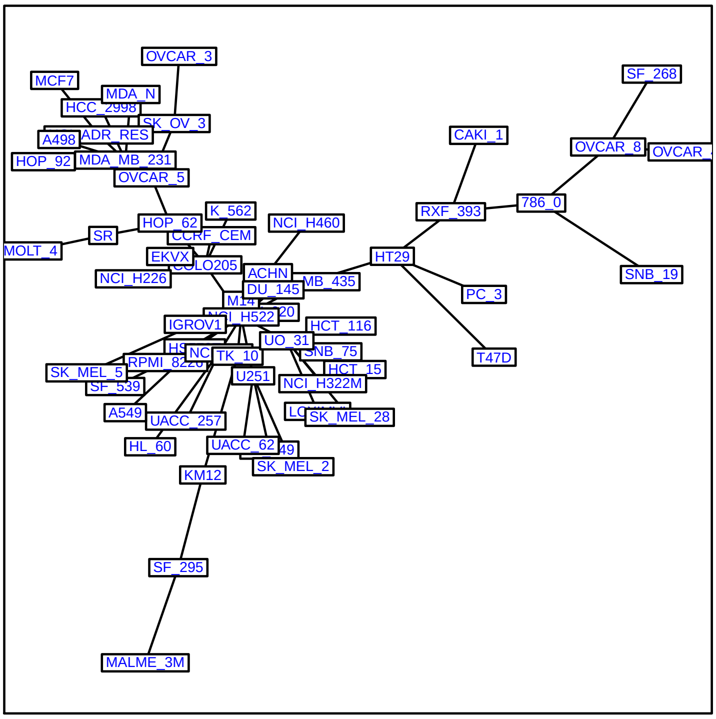


Figure 27: Markov network constructed from maximum weighted spanning tree method on RNA radiation array data [?]. The labels correspond to different cancer cell lines.

against all cell lines, MMCRF is able to tackle this problem better than SVM.

6.4.4 Effect of loopy belief propagation

We also tested different loopy belief propagation iteration parameters to see their effects on global optimization. In Figure 30, we report global objective values under different iteration parameters against time line. The best loopy belief propagation iteration parameter from the experiment was 11, where objective value of MMCRF approached a global optimum in a short time. Smaller iteration parameter is not suf-

Dataset version	***p-value from sign test***	
	Accuracy	***F1 score***
Full	0.6	0.009
Non-Zero-Active	0.04	3.0e-5
Middle-Active	1.2e-6	0.018

Table 11: p-values from two tailed sign tests on different versions of NCI-cancer datasets. All p-values are statistically significant (p-value < 0.05), except the one from accuracy comparison on full data.

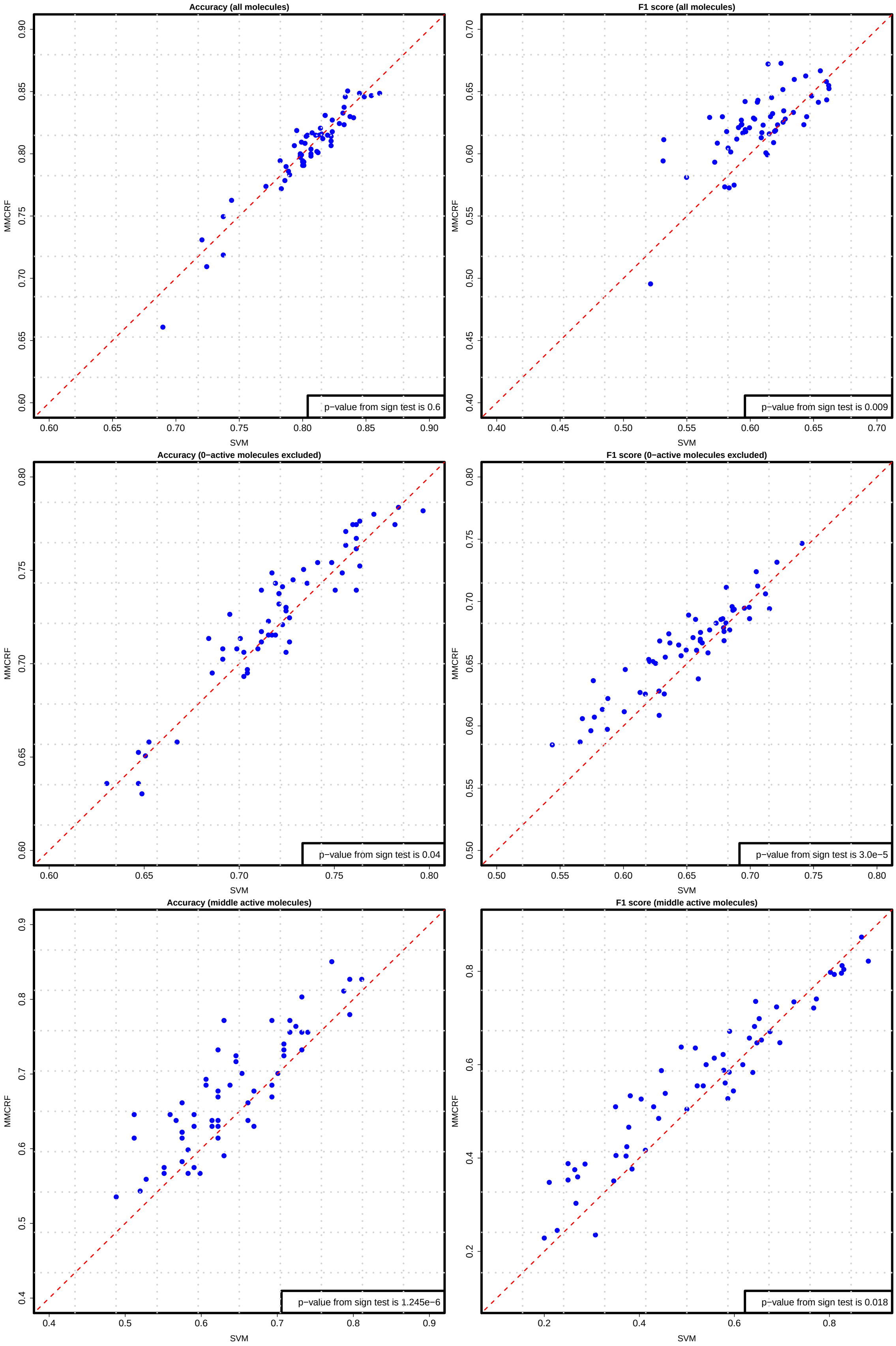


Figure 28: Scatter plot of accuracy (left) and F1 score (right) of MMCRF vs. SVM on Full data (top), No-Zero-Active (middle) and Middle-Active molecules (bottom).

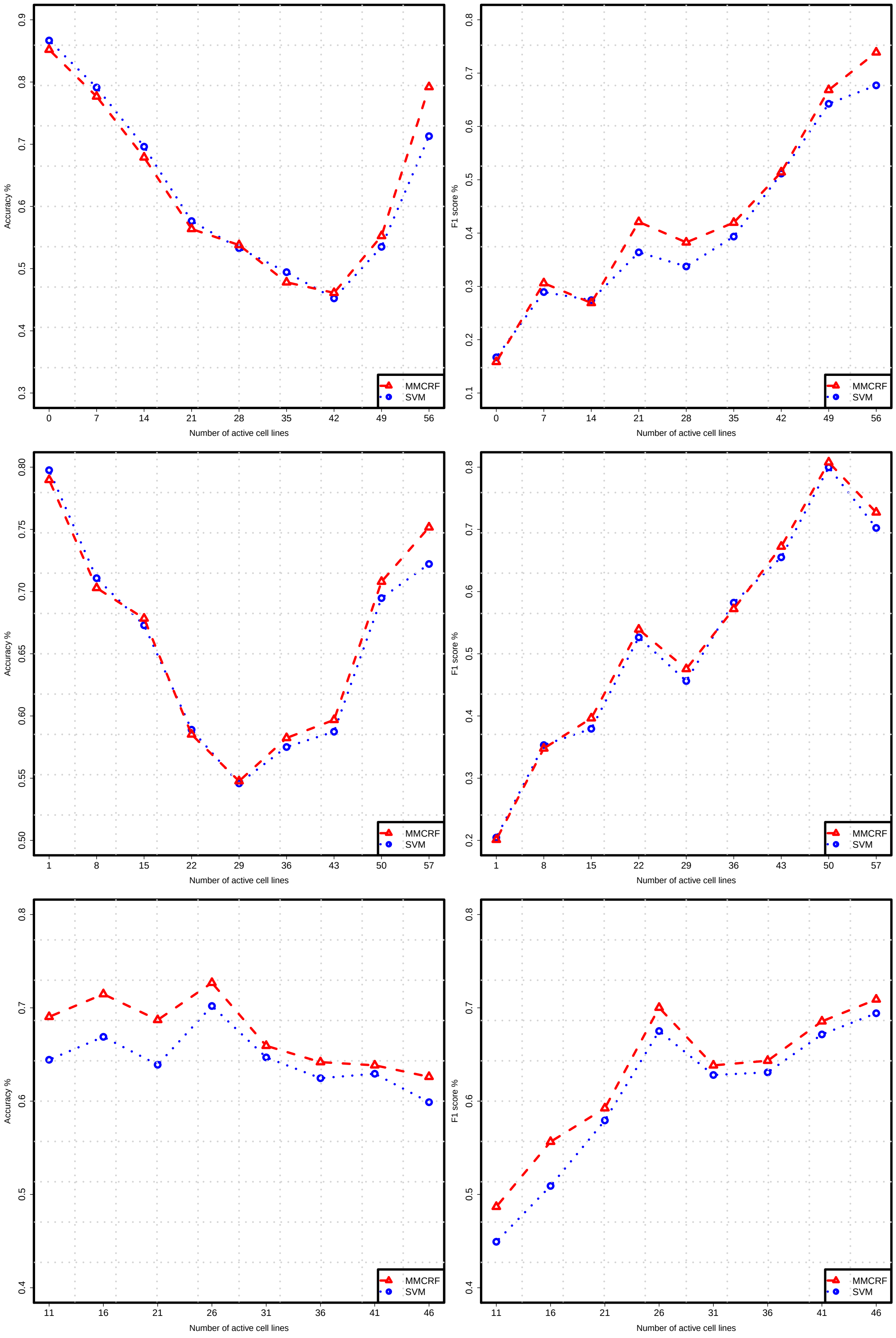


Figure 29: Accuracy (left) and F1 score (right) of MMCRF vs. SVM on Full data (top), No-Zero-Active (middle) and Middle-Active molecules (bottom). Molecules are grouped according to their activities in the number of celllines, denoted as x-axis.

	Positive class		Negative class	
	SVM Correct	SVM Incorrect	SVM Correct	SVM Incorrect
MMCRF Correct	48.6 ± 4.1%	7.1 ± 2.6%	88.0 ± 4.9%	2.2 ± 1.2%
MMCRF Incorrect	3.4 ± 1.3%	40.9 ± 3.4%	3.8 ± 1.7%	6.1 ± 3.0%

Table 12: Agreements of positive and negative predictions from both MMCRF and SVM on Full data.

	Positive class		Negative class	
	SVM Correct	SVM Incorrect	SVM Correct	SVM Incorrect
MMCRF Correct	57.9 ± 5.7%	5.2 ± 1.7%	74.4 ± 10.5%	3.4 ± 1.2%
MMCRF Incorrect	2.8 ± 1.1%	34.1 ± 5.3%	4.4 ± 1.2%	17.8 ± 8.9%

Table 13: Agreements of positive and negative predictions from both MMCRF and SVM on dataset without zero-active molecules.

ficient for MMCRF to reach a global optimum, while larger values incur convergence to need more time. The optimal loopy belief propagation iteration parameter turns out to be close to the diameter of the Markov network (10 in this case), indicating that propagation of messages through the network is required for best performance.

6.4.5 Computation time

Besides predictive accuracy, training time of classifiers is important when a large number of drug targets need to be processed. The potential benefit of multilabel classification is the fact that only single model needs to be trained instead of a bag of binary classifiers. We compared the running time needed to construct MMCRF classifier (implemented in native MATLAB) against libSVM classifier (C++). We conducted the experiment on a 2.0GHz computer with 8GB memory. Figure 31 shows that MMCRF scales better when training set increases.

	Positive class		Negative class	
	SVM Correct	SVM Incorrect	SVM Correct	SVM Incorrect
MMCRF Correct	41.5 ± 25.9%	7.5 ± 5.9%	57.7 ± 26.6%	9.8 ± 6.9%
MMCRF Incorrect	8.7 ± 6.9%	42.3 ± 25.1%	5.7 ± 5.6%	26.8 ± 25.9%

Table 14: Agreement of positive and negative predictions from both MMCRF and SVM on middle-active molecules.

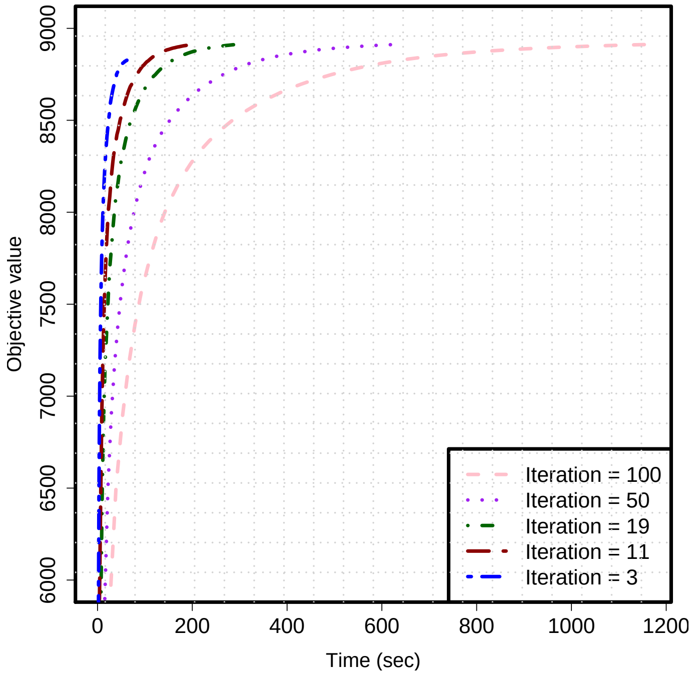


Figure 30: Effect of loopy belief propagation iteration.

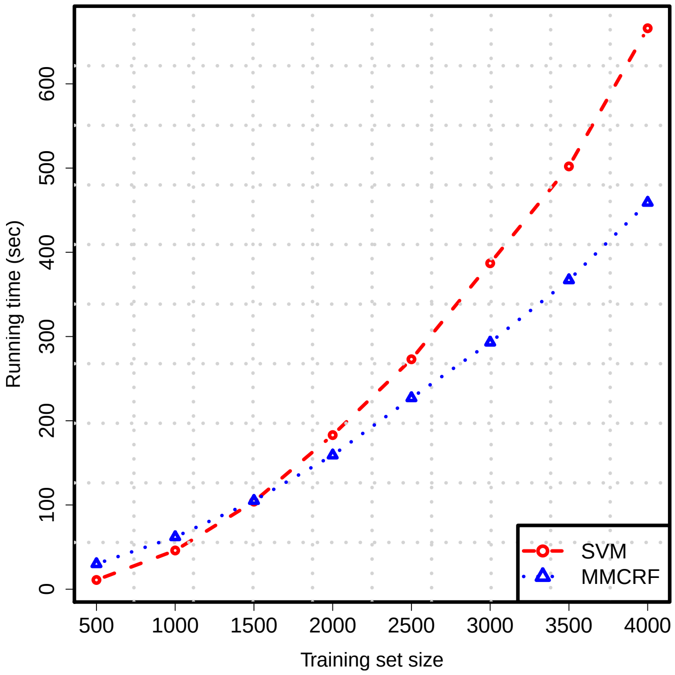


Figure 31: Training time for SVM and MMCRF classifiers on training sets of different sizes.

7 Conclusions

This work explores a machine learning approach for classification of drug-like molecules. Specifically, we focused on two learning algorithms and various graph kernel methods. Support vector machine (SVM) in binary classification scenario and max-margin conditional random field (MMCRF) tailored for multilabel classification task were addressed. We then reviewed and implemented several the state-of-the-art graph kernel methods, and evaluated their performances under binary classification setups. The best kernel method was selected, which achieved top accuracies on Mutag dataset and NCI-cancer dataset of different versions in our experiments.

The multilabel structural classification approach utilized by MMCRF algorithm assumes the dependencies between single labels and predicts all labels in one shot. Significant improvements over SVM classifiers on NCI-cancer dataset were achieved in our experiments by employing multilabel setups. Especially, MMCRF clearly outperformed SVM in one version of NCI-cancer dataset where biased molecules (the ones active against few cell lines or in many cell lines) were excluded.

The performance of MMCRF algorithm was also briefly described in the thesis. The experiment on loopy belief propagation shown that the inference algorithm converged when the iteration parameter was close to the diameter of the underlay network. Hence, the belief had time to propagate from one end of the network to another. The algorithm would not reach a global optimal with a small iteration parameter, and a large parameter only increased the converging time.

The Markov network for modeling the dependencies between the targets can be extracted from auxiliary data with different methods. Currently, only correlation thresholding was considered in our experiments. Other methods, for example graph lasso estimation, still remain mystery to us. Besides, reduced graph which maintains only important structures of original molecular graphs seems to be promising. We need more experiments to find out its effects on predictions. Furthermore, the necessity of the Markov network being generated from auxiliary data is another interesting direction. Comparative experiments are still needed for us to understand the differences between the meaningful networks and random ones. All these questions contribute to a good basis for future work.

References

AHP$^+$08 Astikainen, K., Holm, L., Pitkanen, E., Szedmak, S. and Rousu, J., Towards structured output prediction of enzyme function. *BMC Proceedings*, 2,Suppl 4(2008), page S2.

ASS00 Allwein, E. L., Schapire, R. E. and Singer, Y., Reducing multiclass to binary: A unifying approach for margin classifiers. *Proceedings of 17th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2000, pages 9–16.

BDM$^+$06 Bernazzani, L., Duce, C., Micheli, A., Mollica, V., Sperduti, A., Starita, A. and Tine, M. R., Predicting physical chemical properties of compounds from molecular structures by recursive neural networks. *Chemical Information and Modeling*, 46,5(2006), pages 2030–2042.

BGG$^+$03 Barker, E. J., Gardiner, E. J., Gillet, V. J., Kitts, P. and Morris, J., Further development of reduced graphs for identifying bioactive compounds. *Chemical Information and Computer Sciences*, 43,2(2003), pages 346–356.

BGV92 Boser, B. E., Guyon, I. M. and Vapnik, V. N., A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pages 144–152.

BL03 Burred, J. J. and Lerch, A., A hierarchical approach to automatic musical genre classification. *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)*, september 2003, pages 8–11.

BST06 Barutcuoglu, Z., Schapire, R. E. and Troyanskaya, O. G., Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22,7(2006), pages 830–836.

Bur98 Burges, C. J. C., A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2,2(1998), pages 121–167.

CCF07 Ceroni, A., Costa, F. and Frasconi, P., Classification of small molecules by two-and three-dimensional decomposition kernels. *Bioinformatics*, 23,16(2007), pages 2038–2045.

CST00 Cristianini, N. and Shawe-Taylor, J., *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

CV95 Cortes, C. and Vapnik, V., Support-vector networks. *Machine Learning*, volume 20, 1995, pages 273–297.

DdD$^+$91 Debnath, A. K., de Compardre, R. L., Debnath, G., Shusterman, A. J. and Hansch, C., Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Medicinal Chemistry*, 34,2(1991), pages 786–797.

FHT08 Friedman, J., Hastie, T. and Tibshirani, R., Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9,3(2008), pages 432–441.

FS99 Freund, Y. and Schapire, R. E., Large margin classification using the perceptron algorithm. *Machine Learning*, 37,3(1999), pages 277–296.

Gär03 Gärtner, T., A survey of kernels for structured data. *SIGKDD Explorations*, 5,1(2003), pages 49–58.

Gär05 Gärtner, T., *Kernels for Structured Data.* Ph.D. thesis, University of Bonn, 2005.

GFW03 Gärtner, T., Flach, P. and Wrobel, S., On graph kernels: Hardness results and efficient alternatives. *Lecture notes in computer science*, 2003, pages 129–143.

GM05 Ghamrawi, N. and McCallum, A., Collective multi-label classification. *Proceedings of the 14th ACM International Conference on Information and knowledge management.* ACM, 2005, pages 195–200.

GWB03 Gillet, V. J., Willett, P. and Bradshaw, J., Similarity searching using reduced graphs. *Chemical Information and Modeling*, 43,2(2003), pages 338–345.

GWLB99 Gao, H., Williams, C., Labute, P. and Bajorath, J., Binary quantitative structure activity relationship (qsar) analysis of estrogen receptor ligands. *Chemical Information and Computer Sciences*, 39,1(1999), pages 164–168.

Hau99 Haussler, D., Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, Computer Science Department, UC Santa Cruz, 1999.

HBP$^+$04 Harper, G., Bravi, G. S., Pickett, S. D., Hussain, J. and Green, D. V. S., The reduced graph descriptor in virtual screening and data-driven clustering of high-throughput screening data. *Chemical Information and Computer Sciences*, 44,6(2004), pages 2145–2156.

Her02 Herbrich, R., *Learning Kernel Classifiers: Theory and Algorithms.* MIT Press, 2002.

Hig96 Higham, N. J., *Accuracy and Stability of Numerical Algorithms.* Society of Industrial and Applied Mathematics, Philadelphia, 1996.

HLZ10 Huang, H., Liu, C.-C. and Zhou, X. J., Bayesian approach to transforming public gene expression repositories into disease diagnosis databases. *Proceedings of the National Academy of Sciences*, 107,15(2010), pages 6823–6828.

HMFM64 Hansch, C., Maloney, P. P., Fujita, T. and Muir, R. M., Correlation of biological activity of phenoxyacetic acids with hammett substituent constants and partition coefficients. *Nature*, 194, pages 178–180.

ITAS94 Ihlenfeldt, W. D., Takahashi, Y., Abe, H. and Sasaki, S., Computation and management of chemical properties in cactvs: An extensible networked approach toward modularity and compatibility. *Chemical Information and Computer Sciences*, 34,1(1994), pages 109–116.

Kar00 Karelson, M., *Molecular Descriptors in QSAR/QSPR.* Wiley-Interscience, 2000.

KMG00 Keseru, G., Molnar, L. and Greiner, I., A neural network based virtual high throughput screening test for the prediction of cns activity. *Combinatorial Chemistry and High Throughput Screening*, 3,2(2000), pages 535–540(6).

KMSS96 King, R., Muggleton, S., Srinivasan, A. and Sternberg, M., Structure-activity relationships derived by machine learning: the use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences*, 93,2(1996), pages 438–442.

Kre99 Krebsel, U. H.-G., Pairwise classification and support vector machines. *Advances in kernel methods support vector learning*, 1999, pages 255–268.

KT50 Kuhn, H. W. and Tucker, A. W., Nonlinear programming. *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability.* University of California Press, Berkeley, CA, USA, 1950, pages 481–492.

KTI03 Kashima, H., Tsuda, K. and Inokuchi, A., Marginalized kernels between labeled graphs. *Proceedings of the 20th International Conference on Machine Learning.* ACM, August 2003, pages 321–328.

LRV^{+}09 Lorenzi, P. L., Reinhold, W. C., Varma, S., Hutchinson, A. A., Pommier, Y., Chanock, S. J. and Weinstein, J. N., DNA fingerprinting of the NCI-60 cell line panel. *Molecular Cancer Therapeutics*, 8,4(2009), pages 713–724.

MCF05 Menchetti, S., Costa, F. and Frasconi, P., Weighted decomposition kernels. *Proceedings of the 22nd International Conference on Machine learning.* ACM, 2005, pages 585–592.

MMR$^+$01 Müller, K.-R., Mika, S., Rätsch, G., Tsuda, K. and Schölkopf, B., An introduction to kernel-based learning algorithms. *IEEE Neural Networks*, 12,2(2001), pages 181–201.

MUA$^+$04 Mahé, P., Ueda, N., Akutsu, T., Perret, J.-L. and Vert, J.-P., Extensions of marginalized graph kernels. *Proceedings of the 21st International Conference on Machine learning.* ACM, 2004, page 70.

Nur95 Nurminen, M., To use or not to use the odds ratio in epidemiologic analysis. *European Epidemiology*, 11,4(1995).

RG03 Ramon, J. and Gärtner, T., Expressivity versus efficiency of graph kernels. *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*, 2003, pages 65–74.

RS09 Ranu, S. and Singh, A. K., Mining statistically significant molecular substructures for efficient molecular classification. *Chemical Information and Modeling*, 49,11(2009), pages 2537–2550.

RSSB05 Ralaivola, L., Swamidass, S., Saigo, H. and Baldi, P., Graph kernels for chemical informatics. *Neural Networks*, 18,8(2005), pages 1093–1110.

RSSST06 Rousu, J., Saunders, C., Szedmak, S. and Shawe-Taylor, J., Kernel-Based Learning of Hierarchical Multilabel Classification Models. *The Machine Learning Research*, 7,2(2006), pages 1601–1626.

RSSST07 Rousu, J., Saunders, C., Szedmak, S. and Shawe-Taylor, J., Efficient algorithms for max-margin structured classification. *Predicting Structured Data.* MIT Press, 2007, pages 105–129.

SCB$^+$05 Swamidass, S. J., Chen, J., Bruand, J., Phung, P., Ralaivola, L. and Baldi, P., Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics*, 21,2(2005), pages i359–i368.

SF10 Silla, C. and Freitas, A., A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 34,2(2010), pages 100–110.

SK09 Shivakumar, P. and Krauthammer, M., Structural similarity assessment for drug sensitivity prediction in cancer. *BMC Bioinformatics*, 10,Suppl 9(2009), page S17.

SRN$^+$07 Shankavaram, U. T., Reinhold, W. C., Nishizuka, S., Major, S., Morita, D., Chary, K. K., Reimers, M. A., Scherf, U., Kahn, A., Dolginow, D., Cossman, J., Kaldjian, E. P., Scudiero, D. A., Petricoin, E., Liotta, L., Lee, J. K. and Weinstein, J. N., Transcript and protein expression profiles of the NCI-60 cancer cell panel: an integromic microarray study. *Molecular Cancer Therapeutics*, 6,3(2007), pages 820–832.

SS02 Schölkopf, B. and Smola, A., *Learning with Kernels.* MIT Press, 2002.

SSTPH06 Szedmak, S., Shawe-Taylor, J. and Parado-Hernandez, E., Learning via linear operators: Maximum margin regression. Technical Report, Pascal Research Reports, 2006.

STC04 Shawe-Taylor, J. and Cristianini, N., *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

Sto03 Stockfisch, T. P., Partially unified multiple property recursive partitioning (pump rp) a new method for predicting and understanding drug selectivity. *Chemical Information and Computer Sciences*, 43,5(2003), pages 1608–1613.

STV04 Schölkopf, B., Tsuda, K. and Vert, J.-P., *Kernel methods in Computational Biology.* MIT Press, 2004.

SVK$^+$09 Shankavaram, U., Varma, S., Kane, D., Sunshine, M., Chary, K., Reinhold, W., Pommier, Y. and Weinstein, J., Cellminer: a relational database and query tool for the nci-60 cancer cell lines. *BMC Genomics*, 10,1(2009), pages 277–282.

TBH01 Trotter, M., Buxton, M. and Holden, S., Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Computers and Chemistry*, 26,2(2001), pages 1–20.

TGK03 Taskar, B., Guestrin, C. and Koller, D., Max-margin markov networks. *Neural Information Processing Systems*, 2003.

THJA04 Tsochantaridis, I., Hofmann, T., Joachims, T. and Altun, Y., Support vector machine learning for interdependent and structured output spaces. *Proceedings of 21th International Conference on Machine Learning*, 2004, pages 823–830.

Vap98 Vapnik, V. N., *Statistical Learning Theory.* Wiley-Interscience, September 1998.

Vap99 Vapnik, V., *The Nature of Statistical Learning Theory (Information Science and Statistics).* Springer, second edition, November 1999.

Wat08 Watson, P., Naive bayes classification using 2d pharmacophore feature triplet vectors. *Chemical Information and Modeling*, 48,1(2008), pages 166–178.

WBD$^+$09 Wang, Y., Bolton, E., Dracheva, S., Karapetyan, K., Shoemaker, B., Suzek, T., Wang, J., Xiao, J., Zhang, J. and Bryant, S., An overview of the pubchem bioassay resource. *Nucleic Acids Research*, 38,2(2009), pages D255–D266.

Weh06 Wehenkel, L., Kernelizing the output of tree-based methods. *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 2006, pages 345–352.

WW99 Weston, J. and Watkins, C., Multi-class support vector machines. *Proceedings ESANN*, 12,2(1999), pages 181–201.

WXS$^+$09 Wang, Y., Xiao, J., Suzek, T., Zhang, J., Wang, J. and Bryant, S., Pubchem: a public information system for analyzing bioactivities of small molecules. *Nucleic Acids Research*, 37,2(2009), pages W623–W633.

XDDC07 Xiao, Z., Dellandréa, E., Dou, W. and Chen, L., Hierarchical Classification of Emotional Speech. Technical Report RR-LIRIS-2007-006, LIRIS UMR 5205 CNRS and INSA de Lyon and Université Claude Bernard Lyon and Université Lumière Lyon and Ecole Centrale de Lyon, March 2007.

XLY$^+$04 Xue, Y., Li, Z., Yap, C., Sun, L., Chen, X. and Chen, Y., Effect of molecular descriptor feature selection in support vector machine classification of pharmacokinetic and toxicological properties of chemical agents. *Chemical Information and Modeling*, 44,5(2004), pages 1630–1638.