Structured Data

Structured Output Prediction of Anti-Cancer Drug Activity

Anas ATMANI : anas.atmani@ensae-paristech.fr
Domitille COULOMB : domitille.coulomb@eleves.enpc.fr
Benoît CHOFFIN : benoit.choffin@ensae-paristech.fr
Paul ROUJANSKY : paul.roujansky@ensae-paristech.fr

April 2017

1 Introduction

In our Structured Data project, we decided to work on the article *Structured Output Prediction of Anti-Cancer Drug Activity*[SHR10] (referred to as "the article" in what follows), written by Hongyu Su, Markus Heinonen, and Juho Rousu (Department of Computer Science, Aalto University).

The use of computational tools is very common today in drug discovery, and Machine Learning methods are considered as very effective, due to the complexity and non-linearity of biological systems. These methods are used in several steps of the drug discovery methodology, such as prediction of target structure, prediction of biological activity. The main goal of the article is to predict, with sufficient accuracy, which drug will be the most effective against a given cancer cell line, without having to test hundreds of different molecules in real life.

In the last decades, SVMs and Artificial Neural Networks have been the most efficient models in this regard. Notably, they have enabled to deliver significant results. While these methods tend to be adapted when there is a single output variable, they do not benefit from any structure that may underlie multiple output variables, such as cancer cell lines in our case.

The article proposes a binary classification algorithm that makes use of latent structure amongst both input and output variables. This algorithm takes as input a description of a molecule, and predicts the activity against a range of cancer cell lines at once. Statistical dependencies between the cancer cell lines are encoded by a Markov network where nodes represent cell lines and edges represent similarity according to an auxiliary dataset. Statistical dependencies between the molecules are equally represented through the Gram matrix of a given adapted kernel. We expect this model to outperform the state-of-the-art SVM model.

In our project, we have implemented both a SVM classifier, as well as a Maximum-Margin Conditional Random Field (MMCRF) classifier, and compared their respective performances against a common dataset.

2 Data Preprocessing and Methods

We got the preprocessed data from Mr J. Rousu personal Github page[1]. We have at our disposal 2305 molecules and their individual activity scores against 59 distinct cancer cell lines[2].

2.1 Molecules

The activity score (referred to as "Score") is computed as: Score $= -10 * \log(\text{GI50})$, where GI50 is the drug concentration causing 50% inhibition of the desired activity. However, the activity outcome needs to be a binary value for us to be able to perform a classification task. According to NCBI, we can consider a molecule to be "active" if the activity score is greater or equal to 60 - as a quick note, this is the threshold chosen by Rousu et al. in their article as well([SHR10]). We obtain the following distributions of respectively molecular (continuous) and binary ("active" vs. "inactive") activities:

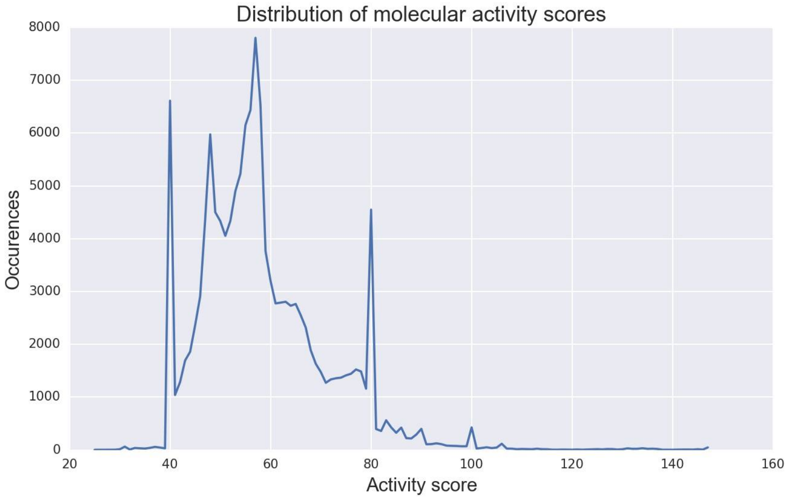

Figure 1: Distribution of molecular activity distribution

[1] `https://github.com/hongyusu/Molecular_Classification`

[2] "cell line": *a cell culture that is derived from one cell or set of cells of the same type and in which under certain conditions the cells proliferate indefinitely in the laboratory.* - The American Heritage® Medical Dictionary Copyright © 2007, 2004 Published by Houghton Mifflin Company. All rights reserved.

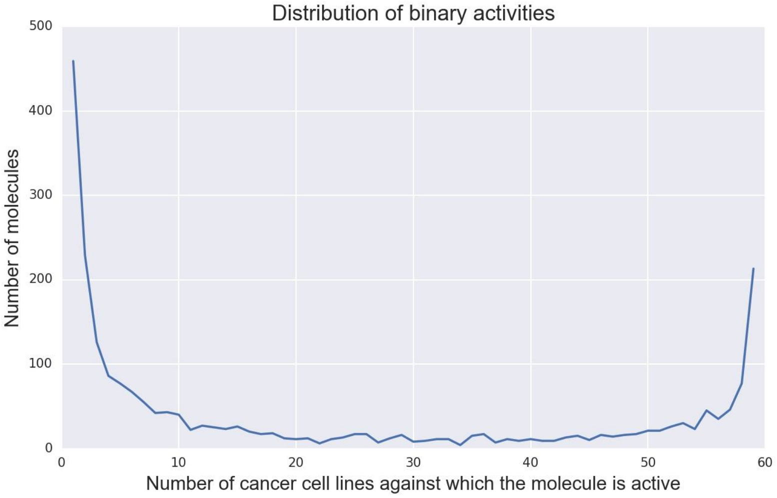

Figure 2: Distribution of binary activity

We observe 3 different peaks of molecular activity at 40, 60 and 80. Our convention will therefore be that when the activity is below 60, the molecule is inactive on the cancer cell, and when it is at 60 and above there is a significant activity.

We can see from figures 1 and 2 that molecular activity data are highly biased over the cell lines. Most of the molecules are inactive in all cell lines, while a relatively large proportion of molecules (10% approximately) are active against almost all cell lines. In the article the authors consider these as being potentially toxic and therefore less likely to be potential drug candidates than the ones in the middle part of the histogram.

2.2 Markov network generation for cancer cell lines

We tested two different methods for creating the Markov network of cancers, which represents the *structure* of the output, i.e. the statistical links between cancers.

- The first method consisted in extracting the network from the correlation matrix between the cancer cell lines, computed from the "target" matrix of response for each molecule, and defining a *link* between two cell lines only if the correlation was above a certain threshold in absolute value. The algorithm used is detailed below in 1.

```
Algorithm 1: Correlation thresholding
Input: correlation matrix $X_c$, size $N \times N$ ; threshold $\tau$
Create empty graph $G$.
for $i = 1, 2, ..., N$ do
    for $j \in \{1, 2, ..., N\} \backslash \{i\}$ do
        if $|X_c[i, j]| > \tau$ then
            Add $(i, j)$ to $G$.
        end
    end
end
Output: Markov network (graph $G$)
```

A fairly simple heuristic to choose the optimal threshold τ was to build a graph sufficiently connected in order to learn from the latent affinities between cancers, but with relatively few connections as well on the other hand in order not to fall in a computational bottleneck. The only drawback from using such a method is that we have no guarantee that a node cannot be isolated from the resulting graph.

- The second method we harnessed to build the Markov network was using the maximum-weight spanning tree (MST)[3] algorithm applied on the correlation matrix of cancer cell lines. The latter was computed on an external matrix of features for each cancer based on RNA-based dataset spanning our cancer cell lines of interest (the "RNA: microRNA OSU V3 chip", available on the National Cancer Institute (NCI) database[4]). The MST algorithm basically minimizes the number of edges and maximizes their weights, under the constraint that the network is **fully connected** – which means that between two random nodes, there will always be a path that can join them. Here-below, the input graph A is computed from the absolute-valued correlation matrix, which can be seen as an affinity matrix.

Algorithm 2: Maximum-weight spanning tree algorithm – Kruskal's algorithm

```
Input: graph $A$
Create empty graph $G$.
foreach $u \in A$.nodes() do
    MAKE-SET($u$)
end
foreach $(u, v) \in A$.edges() ordered by weight $w_{u,v}$, decreasing do
    if FIND-SET($u$) $\neq$ FIND-SET($v$) : then
        $G = G \cup \{(u, v)\}$
        UNION($u$,$v$)
    end
end
Output: Markov network (graph $G$)
```

The complexity of Kruskal's algorithm [Kru56] (above) is in $\mathcal{O}(e \log(n)$ where $e = |G.\text{edges}()|$ and $n = |G.\text{nodes}|$.

As a quick comparison, we obtained the following statistics for both methods, using $\tau = 0.9$ in the case of the first one:

Table 1: Comparison of the two different methods that were performed to build the Markov network for cancer cell lines

	Correlation threshold	**Maximum weight spanning tree**
Number of subgraphs	3	1
Number of isolated nodes	2	0
Number of edges	847	58

We see that the MST algorithm (second method) gives quite a good output since the resulting graph is fully connected with a minimal number of edges – since it is equal to the number of nodes (59) minus

[3] A Python implementation of Kruskal's algorithm for building the maximum-weight spanning tree can be found in `spanningtree.py`, which we outsourced from the `NetworkX` library, available at this address: `https://networkx.readthedocs.io/en/latest/_modules/networkx/algorithms/tree/mst.html`

[4] `https://discover.nci.nih.gov/cellminer/loadDownload.do`

one. We might potentially benefit from such a structure from a computational point of view but perhaps also miss some closer latent links that might exist between cancer cell lines since the algorithm tends to be *non-exhaustive* in terms of connections. On the other hand, the graph obtained from the first method is much more connected, but contains 2 isolated nodes which might – or might not in the end – be completely different from all the other cancers, and therefore not be linked to any of them in the prediction part.

2.3 Summary

Input: We consider 2305 distinct molecules. Each of them has several physico-chemical and geometric properties that enable to build similarities between all molecules through a kernel. We end up with the $[2305 \times 2305]$ Gram matrix of the *Tanimoto kernel.* This last kernel can be simply defined as the ratio between the number of elements of the intersection of the two sets of paths (in that case, we consider each graph as a bag-of-paths) and the number of elements of the union of the two sets. According to Ralaivola et al.[Ral+05], this graph kernel is highly adapted for chemical molecules and yields state-of-the art results on several datasets – including NCI – when tackling our very classification task.

Ouput: We have a total of 59 cancer cell lines for which we would like to predict the effect of each molecule (active/inactive). This last information is provided in a $[2305 \times 59]$ "target" matrix, where "active" and "inactive" states are respectively encoded as +1 and -1. We also have external RNA-based data for each cancer cell line. We can build a similarity graph between all cancer cell lines through either a thresholded correlation matrix or a *maximum weight spanning tree* (MST) algorithm. As a quick note, the graph should not necessarily be fully-connected.

3 Learning algorithm

We tested two different modeling approaches:

- Perform a prediction independently for each cancer cell line, through a standard classification algorithm such as SVM.
- Take into account the similarities between the cancer cell lines and make use of this "structure" (goal of the article) through a MMCRF algorithm.

3.1 First Method: SVM

SVMs (Support Vector Machines) are a very popular class of learning algorithms, which are suited for classification tasks, and are considered today as one of the best keys-in-hand tools for the prediction of the effect of a molecule on a specific cell or virus. Very briefly, the idea behind the method is to build a hyperplane, or a set of hyperplanes, in a high-dimensional space. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. Usually, in order to avoid high computational cost, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a particular kernel function $K(x, y)$ selected to suit the problem.

Figure 3 below shows how a two-dimensional SVM is set up:

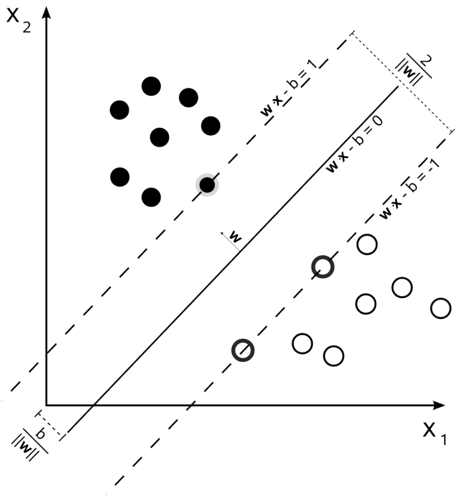

Figure 3: Support Vector Machines Framework

This is therefore equivalent to solving the following problem in the primal space:

$$(\mathcal{P}_{\text{SVM}}) : \begin{cases} \min\limits_{(w,b,\xi)} & \frac{1}{2}\|w\|_2^2 + C\sum_i \xi_i \\ \text{s.t.} & y_i(w^\top x_i + b) \geq 1 - \xi_i \\ \text{and} & -\xi_i \leq 0 \end{cases}$$

where C is a hyperparameter of the model. This is equivalent to the following reformulation in the dual space, using the kernel trick:

$$(\mathcal{D}_{\text{SVM}}) : \begin{cases} \underset{\mu}{min} & \frac{1}{2}\left(\sum_{i=1}^n \sum_{j=1}^n \mu_i \mu_j y_i y_j K(x_i, x_j)\right) - \mu^\top u \\ \text{s.t.} & y^\top \mu = 0 \\ \text{and} & 0 \leq \mu \leq C \end{cases}$$

where K is a positive definite kernel.

We implemented the SVM algorithm in Python; it can be found in notebooks `SVM.ipynb`. For this purpose, we partly used the `sklearn` library – very useful to perform Machine Learning tasks – and the traditional fast-computing library `numpy`.

3.2 Second Method: Max-margin Conditional Random Field (MMCRF)

The MMCRF framework consists in the following components:

- Max-margin learning: Maximize the margin between real example $\phi(x_i, y_i)$ and all the incorrect pseudo-examples $\phi(x_i$, y), whilst controlling the norm of the weight vector;
- Use of the kernel trick, with K a positive definite kernel, to tackle high-dimensionality of input feature maps. Various kernels have been tested in the literature, and Tanimoto Kernel seems to be among the working best, as previously explained (see [Ral+05]). We have here directly at our disposal the Gram matrix as an input;
- Use of graphical model techniques to tackle the exponential size of the multilabel space.
  - Marginal dual representation to obtain polynomial size (dual) variable set;

- Probabilistic inference (loopy belief propagation) over the marginal dual polytope to give fast updates during optimization.

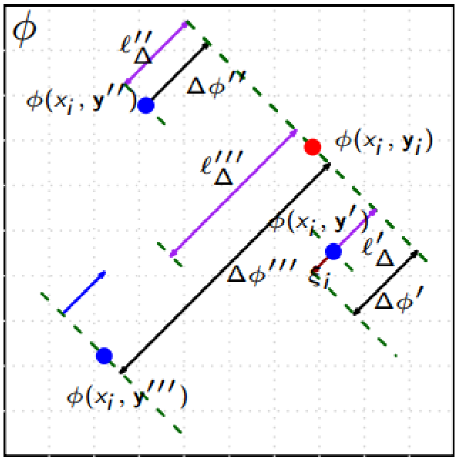

Figure 4: MMCRF Overview

More specifically, the MMCRF learning algorithm takes as input a kernel matrix and a label matrix containing the multilabels of the training patterns. Moreover, we assume that there is an associative network $G = (V, E)$ between the ouput variables - here the cancer cell lines - where node $j \in V$ corresponds to the j^{th} component of the multilabel and the edges $e = (j, j_0) \in E$ correspond to a microlabel dependency structure. The model takes the form of a conditional random field with exponential edge-potentials:

$$\text{MMCRF model:} \quad P(y|x) \propto \prod_{e \in E} \exp\left(\mathbf{w}_e^\top \varphi_e(x, \mathbf{y}_e)\right) = \exp\left(\mathbf{w}^\top \varphi(x, \mathbf{y})\right)$$

Where $\mathbf{y}_e = (y_j, y_{j_0})$ denotes the pair of microlabels of the edge $e = (j, j_0)$; $\mathbf{y}_e$ can take 4 possible different values, from $\mathcal{Y} = \{[-1,-1],[-1,+1],[+1,-1],[+1,+1]\}$, depending on the binary adjacent node labels. A joint feature map $\varphi_e(x, \mathbf{y}) = \phi(x) \otimes \psi_e(\mathbf{y}_e)$ for an edge is composed via tensor product of input $\phi(x)$ and output feature map $\psi(\mathbf{y})$ – where $\psi_e(\mathbf{y}) = [\psi_e^u(\mathbf{y})]_u = [[\![\mathbf{y}_e = u]\!]]_u$ – thus including all pairs of input and output features. The parameters are learned by maximizing the minimum loss-scaled margin between the correct training examples $(x_i, \mathbf{y}_i)$ and incorrect pseudo-examples $(x_i, \mathbf{y})$; $\mathbf{y} \neq \mathbf{y}_i$, while controlling the norm of the weight vector. The resulting primal soft-margin optimization problem is the following:

$$\text{MMCRF primal soft-margin :} \begin{cases} \min\limits_{(w, \xi \leq 0)} & \frac{1}{2}\left\|w\right\|_2^2 + C\sum_{i=1}^m \xi_i \\ \text{s.t.} & \mathbf{w}^\top(\varphi(x_i, \mathbf{y}_i) - \varphi(x_i, \mathbf{y})) \geq \ell(\mathbf{y}_i, \mathbf{y}) - \xi_i \quad \text{, for all } i \text{ and } \mathbf{y} \end{cases}$$

which is equivalent to the following reformulation in the dual space:

$$\text{MMCRF dual soft-margin :} \begin{cases} \max\limits_{\alpha \geq 0} & \alpha^T\gamma - \frac{1}{2}\alpha^T K \alpha \\ \text{s.t.} & \sum_{\mathbf{y}} \alpha(i, \mathbf{y}) \geq C \quad \forall i, \mathbf{y} \end{cases}$$

where $K = \Delta\mathbf{\Phi}^\top\mathbf{\Phi}$ is the *joint* kernel matrix (computed from $\varphi(.,.)$) for pseudo-examples $(x_i, \mathbf{y})$ and $\gamma = \gamma(\mathbf{y}_i, \mathbf{y}))_{i,\mathbf{y}}$ encodes the margin requirements for each $(x_i, \mathbf{y})$.

Because there are exponentially many dual variables $\alpha(i, \mathbf{y})$, one for each pseudo-example, this dual problem is exponentially complex, and is translated to a polynomially-sized form by considering the marginals of the dual variables only: this method is called "marginal dual method".

The feasible set of problem (MMCRF dual soft-margin) is a cartesian product $\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_m$ of identical closed polytopes:

$$\mathcal{A}_i = \{\alpha_i \in \mathbb{R}^{|\mathcal{Y}|} : \alpha_i \geq 0, \|\alpha_i\|_1 \leq C\}$$

where $|\mathcal{Y}| = 4$ in this case. We then define the *marginal polytope* of α_i on H as:

$$\mathcal{M}_i = \{\mu_i \,|\, \exists \alpha_i \in \mathcal{A}_i : M_H \alpha_i = \mu_i\}$$

where M_H is the matrix given by $M_H(e, \mathbf{u}_e; \mathbf{y}) = [\mathbf{u} = \mathbf{y}_e]$ and μ_i is the marginal dual vector of the example x_i such that $\mu_i = (\mu(i, e, \mathbf{u}_e)_{e \in E, \mathbf{u}_e \in \mathcal{Y}_e}$.
Theorem 1 from [Rou+07] gives that "the feasible set of the marginalized problem is the marginal dual polytope, or to be exact the cartesian product of the marginal polytopes of single examples (which are in fact equal):

$$\mathcal{M} = \mathcal{M}_1 \times ... \times \mathcal{M}_m\text{”}$$

From [Rou+07], we obtain that, given an orthogonal feature representation inducing a decomposable kernel, the quadratic part of the objective becomes:

$$\alpha K \alpha = \mu^\top K_H \mu$$

where $K_H = \operatorname{diag}(K_e, e \in E)$ is a block diagonal matrix with hyperedge-specific kernel blocks K_e. The marginalized optimization problem can be stated in implicit form as:

$$\text{MMCRF marginalized formulation :} \quad \max_{\mu \in \mathcal{M}} \mu^\top \ell_H - \frac{1}{2} \mu^\top K_H \mu$$

where ℓ_H is the chosen loss and $\mu = (\mu_i)_{i=1,...,m}$ is the marginal vector of the whole training set, consisting in the concatenation of the single example marginal dual vectors.

In order to measure the inaccuracy of our predictions, we use the Hamming loss which will gradually increase with the number of incorrect microlabels.

$$\text{Hamming loss:} \quad \ell_\Delta(\mathbf{y}, \mathbf{u}) = \sum_j [\![y_j \neq u_j]\!]$$

As a quick example, the Hamming loss for two observations $\mathbf{y} = [-1, -1]$ and $\mathbf{u} = [+1, +1]$ is computed as the length of the shortest path (here equal to 2) from one node to another in the following hypercube graph (which, in this case, is simply a square), which is equivalent to computing the number of distinct microlabels between $\mathbf{y}$ and $\mathbf{u}$ (here 2 therefore).

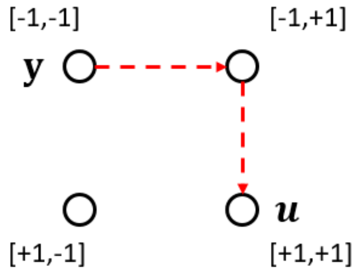

Figure 5: Example of the Hamming loss calculation.

The below algorithms 3 and 4, reproduced from [Rou+07], detail how problem (MMCRF marginalized formulation) – which is a quadratic program – is solved:

Algorithm 3: Maximum margin optimization algorithm for a Conditional Random Field on a hypergraph

```
Input: Training data $S = ((\mathrm{x}_i, \mathrm{y}_i))_{i=1}^{m}$, hyperedge set $E$ of the hypergraph, a loss vector $l$, and the
       feasability domain $\mathcal{M}$
Initialize $\mathbf{g} = l_\mu$, $\xi = l$, $dg = \infty$ and $OBJ = 0$.
while $dg > dg_{min}$ & $iter < max\_iter$ do
    $[WS, Freq] = \mathrm{UpdateWorkingSet}(\mu, \mathbf{g}, \xi)$;
    Compute x-kernel values $K_{X,WS}$ with respect to the working set;
    for $i \in WS$ do
        Compute joint kernel block $K_{ii}$ and subspace gradient $\mathrm{g}_i$;
        $[\mu_i, \Delta obj] = \mathrm{CSGA}(\mu_i, \mathrm{g}_i, K_{ii}, \mathcal{M}_i, Freq_i)$;
    end
    Compute gradient g, slacks $\xi$ and duality gap $dg$;
end
Output: Dual variable vector $\mu$ and objective value $f(\mu)$
```

Algorithm 4: Conditional subspace gradient ascent optimization step (CSGA)

```
$\mathrm{CSGA}(\mu_i, \mathrm{g}_i, K_{ii}, \mathcal{M}_i, maxiter_i)$
Input: Initial dual variable vector $\mu_i$, gradient $\mathrm{g}_i$, the feasible region $\mathcal{M}_i$, a joint kernel block $K_{ii}$
       for the subspace, and an iteration limit $maxiter_i$
$\Delta obj = 0; iter = 0$
while $iter < maxiter$ do
    # find highest feasible point given $\mathrm{g}_i$
    $\mu^* = \mathrm{argmax}_{\mathrm{v} \in \mathcal{M}_i} \mathrm{g}_i^T \mathrm{v}$;
    $\Delta\mu = \mu^* - \mu_i$;
    $q = \mathrm{g}_i^T \Delta\mu, r = \Delta\mu^T K_{ii} \Delta\mu$; # taken from the solution of equation (1) below;
    $\tau = \min(\frac{q}{r}, 1)$; # clip to remain feasible
    if $\tau \leq 0$ then
        break; # no progress, stop
    end
    else
        $\mu_i = \mu_i + \tau\Delta\mu$; # update
        $\mathrm{g}_i = \mathrm{g}_i - \tau K_{ii} \Delta\mu$;
        $\Delta obj = \Delta obj + \tau q - \tau^2 \frac{r}{2}$;
    end
    $iter = iter + 1$
end
Output: New values for dual variables $\mu_i$ and change in the objective $\Delta obj$
```

We also implemented the MMCRF algorithm in Python; it can be found in notebook `MMCRF.ipynb`. For this purpose, we studied the Matlab implementation of the MMCRF algorithm from the authors of [SHR10] themselves (the link of their Github page can be found in a footnote above)[5], which we adapted in Python and to our needs. It is basically the implementation of the above algorithms. The notebook

[5] We would like to thank very much MM. Su, Rousu and Heinonen, authors of [SHR10] for their kind help!

contains comments so that the reader can easily understand them.

4 Results and Comparison

We followed [SHR10] and made use of a stratified k-fold (with 5 folds precisely) scheme to compute the metrics on each of our experiments. The folds were stratified using the number of cell lines against which the molecule is active (ranging from 0 to 59 in the original dataset) in order to have the continuum of activity well represented in each fold. To assess the performance of our models, we used mainly two metrics: the accuracy (defined as the number of well classified samples over the total number of samples) and the F1-score – these metrics were computed iteratively on each fold and then averaged. The F1-score may be seen as the harmonic mean of the precision and the recall:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \in [0, 1]$$

with

$$\text{precision} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|} \quad \text{and} \quad \text{recall} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|}$$

TP, FP and FN stand respectively for "True Positive", "False Positive" and "False Negative". A higher value for F_1 means a better "accuracy" for the model.

4.1 Results

We performed the current comparison on the full dataset, with basic hyperparameters (e.g. $C = 100$ as recommended by the authors of the article). The results can be found in table 2.

Table 2: Comparison of the metrics between the network building methods (MMCRF)

	Correlation threshold	**Maximum weight spanning tree**
Accuracy	52.42% ± 1.46%	54.6% ± 2%
F1 score	40.06% ± 2.9%	35.13% ± 1.56%

We remark that there seems to be a trade-off between accuracy and F1 score here: indeed, the correlation threshold method results in a worse accuracy (2 points less) than the maximum weight spanning tree, whereas the effect is opposite for the F1 score (5 points more). This may stem from the fact that the correlation threshold induces a more dense network, yielding more information about the structure of the output. Finally, the network building method does not seem to have any significant impact on the variance of the metrics. We therefore chose to keep the MST algorithm for building the Markov network between every cancer cell lines (second method).

4.2 Different dataset versions

Following the authors of [SHR10], we also compared two versions of the initial dataset: the complete dataset (2305 molecules, 59 cancers; known as *No-Zero-Active* dataset) and a reduced version of this dataset (527 molecules, 59 cancers; known as *Middle-Active* dataset). This last version was built by keeping molecules that were both active and inactive at least against 10 cancers in both case. This made sense to perform analyses on such a dataset because molecules that are effective against less than 10 cancers are less likely to be of interest for effective use; on the contrary, molecules active against more than 49 cancers are very likely to be toxic for the human body and could therefore be removed from the dataset.

As expected, the computing time was much lower (since *Middle-Active* only represents 23% of *No-Zero-Active*) during our experiments for the reduced version of the dataset. We compared the performances of the two versions of the dataset with both of our models and obtained the results in tables 3 and 4 below. We performed our comparisons with a C value of 100 for both the SVM and the MMCRF.

Table 3: Comparison of the overall accuracies between the dataset versions

	No-Zero-Active	**Middle-Active**
SVM	60.76% ± 4.07%	58.3% ± 7.97%
MMCRF	54.60% ± 2%	50.8% ± 1.68 %

Table 4: Comparison of the F1 score between the dataset versions

	No-Zero-Active	**Middle-Active**
SVM	43.84% ± 6.52%	46.03% ± 19.48%
MMCRF	35.13% ± 1.56%	44.15% ± 3.14%

We observe here that the SVM accuracy is globally better on the No-Zero-Active dataset, but its F1-score is worse on this dataset. It can be easily explained by the fact that for the No-Zero-Active dataset, there are plenty of molecules that are either active against a lot of cell lines, or against a very few cell lines. As a consequence, we can imagine that the SVM underfits for this dataset, and for these particular molecules, predicts always "Active" or "Inactive". This can result in an overall better accuracy, but also a lesser F1-score because this metric takes by definition also into account the number of False Negatives and False Positives. Let us note however that the SVM seems to be less stable on the Middle-Active dataset (this may be the effect of its size).

As for the MMCRF, it is crucial to first take into account that for computational matters, the present comparison was not performed using optimal hyperparameters; the goal was more to compare the effect of each dataset on *each* model than to compare the performances *between* the models. The results in 3 and 4 for MMCRF are as a consequence not representative of the real power of this algorithm. Here, the effects seem to be similar to SVM. Nonetheless, the variance of each metric is much lower, no matter the dataset.

4.3 Comparison of the two models

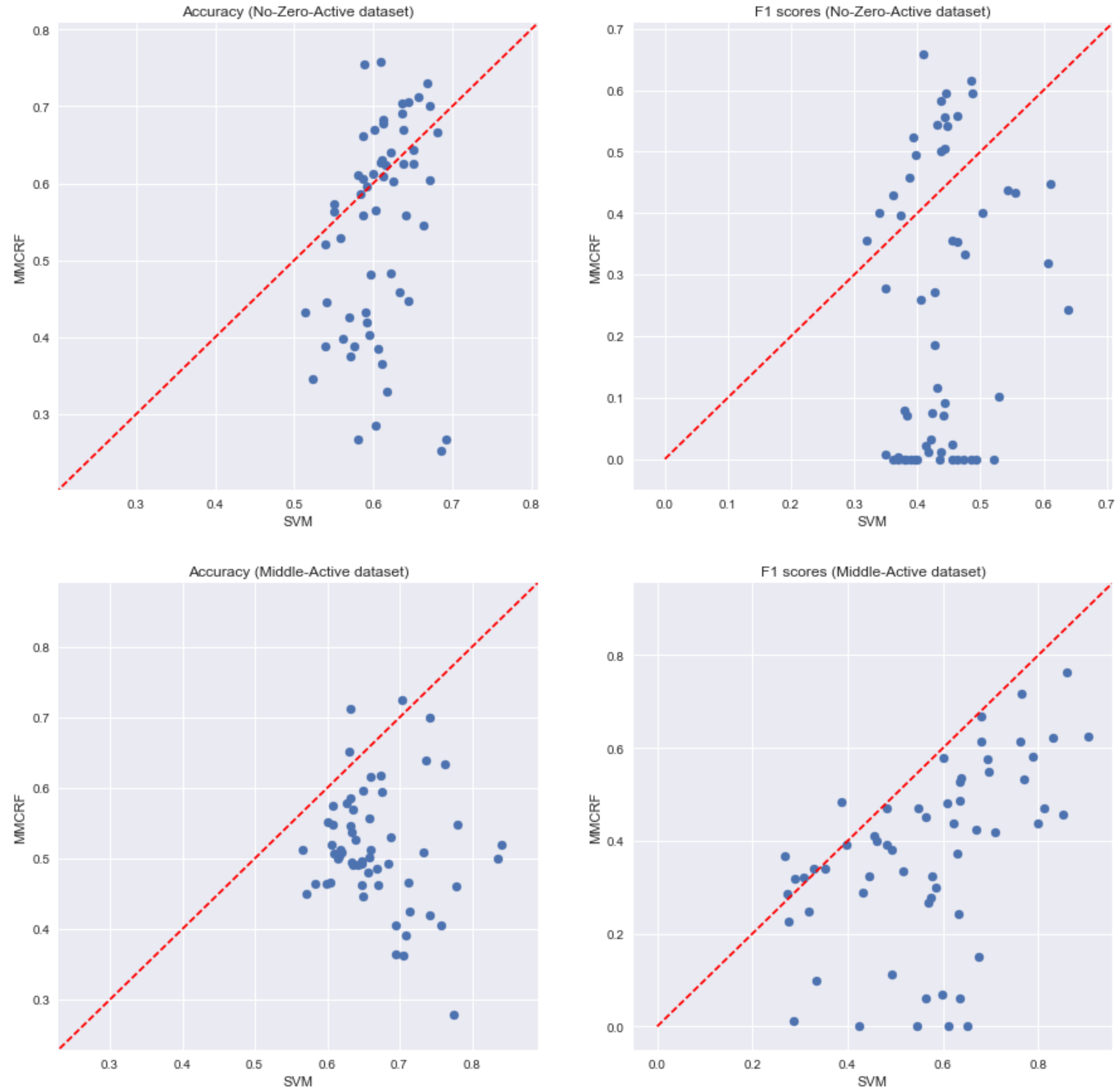

Figure 6: Comparison of the metrics of the two models on each cancer cell line and each dataset

Table 5: Comparison of the predictions of the two models on the positive class (activity)

	SVM correct	**SVM incorrect**
MMCRF correct	14.6%	18.3%
MMCRF incorrect	30.1%	37.0%

Table 6: Comparison of the predictions of the two models on the negative class (no activity)

	SVM correct	**SVM incorrect**
MMCRF correct	46.8%	19.4%
MMCRF incorrect	23.1%	10.7%

5 Conclusion

In this project, we managed to successfully implement a structured output prediction approach for the classification of drug-like molecules. This method enables to use latent statistical dependencies between multiple output variables by means of a Markov network built on auxiliary data. We implemented as well a state of the art SVM classifier, and showed that the MMCRF takes more into account the structural dependencies of the input and the output, even if our experiments, for computational matters, did not really yield any information about the true performance of the MMCRF algorithm – for instance, we were not able to perform any effective gridsearch method to optimize the hyperparameters of the model.

References

[Kru56] Joseph B Kruskal. “On the shortest spanning subtree of a graph and the traveling salesman problem”. In: *Proceedings of the American Mathematical society* 7.1 (1956), pp. 48–50.

[Ral+05] Liva Ralaivola et al. “Graph kernels for chemical informatics”. In: *Neural networks* 18.8 (2005), pp. 1093–1110.

[Rou+07] Juho Rousu et al. “Efficient algorithms for max-margin structured classification”. In: *Predicting structured data* (2007), pp. 105–129.

[SHR10] Hongyu Su, Markus Heinonen, and Juho Rousu. “Structured output prediction of anti-cancer drug activity”. In: (2010), pp. 38–49.