Identification of genomic markers for drug sensitivity using Multi-task Learning

Master thesis

Thesis committee
Prof.dr. L.F.A. Wessels
Dr.ir. J. de Ridder
Prof.dr. J.J. Goeman
Dr. S. Canisius

Author	Nanne Aben
Email	nanne.aben@gmail.com
Student	1369342
Thesis supervisors	Dr.ir. J. de Ridder
	Prof. dr. L.F.A. Wessels
Date	January 9, 2014

NKI-AVL

TU Delft Delft University of Technology

Identification of genomic markers for drug sensitivity using Multi-task Learning

Reading guide

Nanne Aben[1,2]

[1]Delft Bioinformatics Lab, Delft University of Technology, 2628 CD Delft, The Netherlands
[2]Bioinformatics and Statistics, Department of Molecular Carcinogenesis, 1066 CX Amsterdam, Netherlands Cancer Institute, The Netherlands

This master thesis consists of three parts. In the first part, I discuss how Multi-task Learning can be used to combine the prediction of drug sensitivity for multiple drugs. The second part contains supplementary information about the first part, but also includes my work on two approaches that I did not further pursue: using Transfer Learning to combine the analyses for different lineages and using Network-constrained Regularization to incorporate prior knowledge on pathways. For the latter, I also propose an enhanced version of the original algorithm. The third part is a log that I have maintained during this project, which summarizes my progress roughly every two weeks, up until the moment I started writing the thesis. Due to its size, I have not included the log in the printed version of this thesis.

Identification of genomic markers for drug sensitivity using Multi-task Learning

Master thesis

Nanne Aben[1,2]

[1]Delft Bioinformatics Lab, Delft University of Technology, 2628 CD Delft, The Netherlands
[2]Bioinformatics and Statistics, Department of Molecular Carcinogenesis, 1066 CX Amsterdam, Netherlands Cancer Institute, The Netherlands

Abstract

Targeted anticancer medicine holds much promise, but determining the optimal treatment for a given patient is often difficult. In order to improve treatment selection, research has focused on the identification of genomic markers of drug sensitivity, often guided by statistical methods that identify relations between characteristics of a tumour's DNA and drug response. However, these statistical methods often result in many spuriously identified relations, which confound the discovery of genomic markers in follow-up experiments. We propose to reduce the amount of spuriously identified relations by using Multi-task Learning, a machine learning technique that identifies relations between DNA and drug response simultaneously for multiple drugs. We show that using Multi-task Learning, the number of identified relations between DNA and drug response can be strongly reduced, while retaining similar prediction performance.

1 Introduction

There are many genetic aberrations that can lead to the development of cancer [15]. A single therapy that effectively treats all these forms of cancer is unlikely. Instead, there has recently been a focus on cancer drugs that selectively inhibit specific aberrant proteins [24]. While this approach has been successful in subsets of patients, the overall clinical response to appropriately selected cancer drugs varies greatly [5,17]. It seems that drug response cannot be fully explained by just the drug target, but instead depends on a complex interplay of various genetic factors. As the mechanisms of this interplay are yet poorly understood, current research focuses on identifying the genetic aberrations involved in this interplay. This should lead to the discovery of genomic markers of drug sensitivity, which can be used to improve the selection of treatment for a given patient.

Numerous studies have used cell lines to study the relation between genetic abberations and drug sensitivity [2, 3, 6, 12, 13, 19, 20, 27–29, 31]. Even though these cell lines lack the complex inter-cellular interactions present *in vivo* and can therefore only approximate drug response in humans, they make excellent model systems for systematic drug sensitivity analysis. Since they divide infinitely, multiple drugs can be tested individually and in different concentrations on the same cell line, resulting in drug sensitivity measurements on genomically identical material. By measuring the size of the cell population that survives in these experiments, the drug response can be quantitatively characterized.

Usually, these studies would be restricted to a specific lineage and a specific subset of anti-cancer drugs. For example, Solit et al. [27] studied drug response to MEK-inhibitors in cell lines carrying a BRAF-mutation and Neve et al. [20] studied drug response sensitivity to Herceptin in breast cancer cell lines. In

two recent studies, Barretina et al. [2] and Garnett et al. [12] have each composed a human tumour cell line drug sensitivity screen of unprecedented size. In the screen of Garnett et al. 661 tumour cell lines have been genetically profiled for mutations, copy number variations and gene expression. Furthermore, for each of these cell lines, they measured the drug response to 139 cancer drugs.

In order to study the relation between the genetic profile and the drug response, Garnett et al. have fitted a statistical model that identifies which genomic features are most important to predict the drug sensitivity. The set of selected genomic features can be used in follow-up experiments to determine genomic markers of drug sensitivity. However, a major problem in this set-up is that the screen is severely undersampled: there are 12.778 genomic features, but only 661 cell lines in which they were measured. Due to the undersampled nature, there will be many features whose correlation with the drug response equals that of genomic features actually causing the drug sensitivity. These features will often also be selected, resulting in spuriously selected features and confounding the identification of genomic markers.

In this work, we propose to address the problem of spuriously selected features by using a machine learning technique called Multi-task Learning [4]. Multi-task Learning was originally proposed to simultaneously learn related problems, such as handwriting recognition from different persons [22]. We propose to use Multi-task Learning to jointly predict the drug response for groups of related drugs. Since the probability that a genomic feature correlates with multiple drug responses by chance is smaller than the probability that a feature correlates with a single drug response by chance, Multi-task Learning can reduce the number of spuriously selected features.

Many algorithms have been proposed for Multi-task Learning. Caruana et al. [4] have proposed a framework using Artificial Neural Networks. Evgeniou and Pontil [9] propose an SVM with additional features for each task. Independently, both Obozinski et al [21, 22] and Evgeniou et al. [1, 8] have proposed to use Group Lasso constraints [32] to tie together the feature selection over various tasks. In the software package Glmnet [11] a variation of this algorithm is proposed, in which the input data is the same for all tasks and only the output varies. This seems appropriate for the drug sensitivity screen, where the genomic measurements are the same between tasks, but the drug sensitivity data does differ.

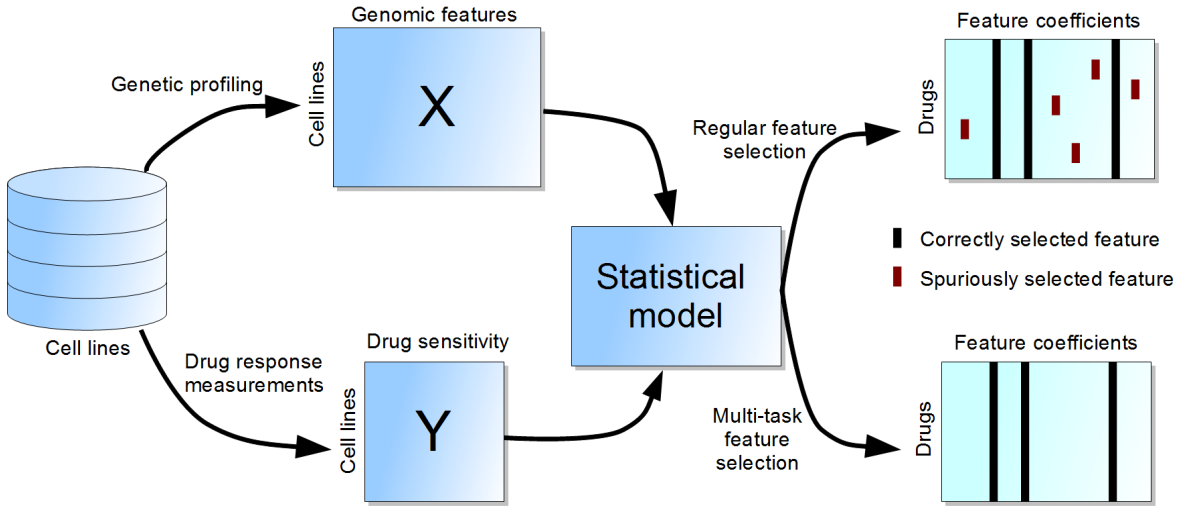


Figure 1: Each cell line in the drug sensitivity screen has been genetically profiled. The resulting measurements are represented in the matrix $\mathbf{X}$. Furthermore, for each cell line, the drug response is measured to 139 anti-cancer drugs. These measurements are represented in the matrix $\mathbf{Y}$. To study the relation between the genomic features and the drug response, a statistical model is fitted to predict $\mathbf{Y}$ from $\mathbf{X}$. Feature selection methods are used to identify which genomic features contribute most to the prediction. We will show that by using Multi-task Learning, the number of spuriously selected features can be reduced, as compared to regular feature selection methods.

In this work, we discuss four Multi-task Learning techniques, including the commonly used technique based on the Group Lasso, but we also propose three novel techniques based on the Sparse Group Lasso and Forward Feature Selection. Using artificial data we show that the Multi-task Learning outperforms regular learning and results in far fewer spuriously selected features. Finally, we apply Multi-task Learning to the drug sensitivity screen and we show that the number of genomic features selected by the statistical model can be greatly reduced, while retaining similar performance.

2 Drug sensitivity screen data

In the drug sensitivity screen of Garnett et al., the relation between $p = 12.778$ genomic features and drug response to $q = 139$ anticancer drugs was studied in $n = 661$ cell lines. Garnett et al. have fitted a statistical model for each drug, defining a $n \times p$ input matrix $\mathbf{X}$ with p genomic features for n cell lines and a $n \times 1$ output vector $\mathbf{y}$ with the drug response of all n cell lines to the drug under investigation. For each drug, this then results in a $1 \times p$ mapping $\boldsymbol{\beta}$ that relates the p measurements in $\mathbf{X}$ to the corresponding drug sensitivity in $\mathbf{y}$. To identify which genomic features contribute most the the prediction of drugs sensitivity, they have applied a feature selection technique which enforces the mapping $\boldsymbol{\beta}$ to have only few non-zero values. The resulting group of selected features could be used in follow-up experiments to determine genomic markers of drug sensitivity.

In this work, we have investigated whether combining the analyses for groups of similar drugs improves the statistical power. Using clustering, we show that the screen contains several groups of drugs that show similar drug response over the cell line panel (Section 4.2.1). We then use Multi-task Learning to jointly perform the feature selection for these groups of similar drugs, defining each task as a drug. We will show that by using Multi-task Learning, the number of spuriously selected features can be reduced, as compared to regular feature selection methods.

3 Methods

We will consider the Group Lasso based technique proposed in [11], as well as novel techniques based on Forward Feature Selection and the Sparse Group Lasso. In Section 3.1, we will discuss Majority Vote Meta-task, a very simple Multi-task Learning technique that converts the problem to a regular, Single-task Learning problem. In Section 3.2 we will discuss the technique based on Forward Feature Selection, for which we will first introduce Regular Forward Feature Selection in Section 3.2.1 and then relate it to Multi-task Forward Feature Selection in Section 3.2.2. Similarly, for the Multi-task Learning based on the Group Lasso and the Sparse Group Lasso discussed in Section 3.3, we first introduce Lasso in Section 3.3.1, we discuss the Group Lasso and Sparse Group Lasso in Section 3.3.2 and then relate those concepts to Multi-task Learning in Section 3.3.3.

3.1 Majority Vote Meta-task

The simplest Multi-task Learning technique that we will consider is Majority Vote Meta-task. Given a $n \times q$ output matrix $\mathbf{Y}$, where each column contains the labels for a certain task, this method creates a single $n \times 1$ meta-task by taking the majority vote over the tasks. This meta-task can then be used in a regular, single-task feature selection and classification scheme. In this work, we will use Majority Vote Meta-task with Regular Forward Feature Selection and Logistic Regression, as described in Section 3.2.1.

3.2 Multi-task learning using Forward Feature Selection

3.2.1 Regular Forward Feature Selection

Feature selection is a technique to identify the subset of features which contribute most to the prediction. Since exhaustive testing of all possible combinations of features is infeasible, heuristics such as Forward Feature Selection need to be used. In Forward Feature Selection, one starts out with an empty set of features and then greedily adds features, such that in each round the feature contributing the most

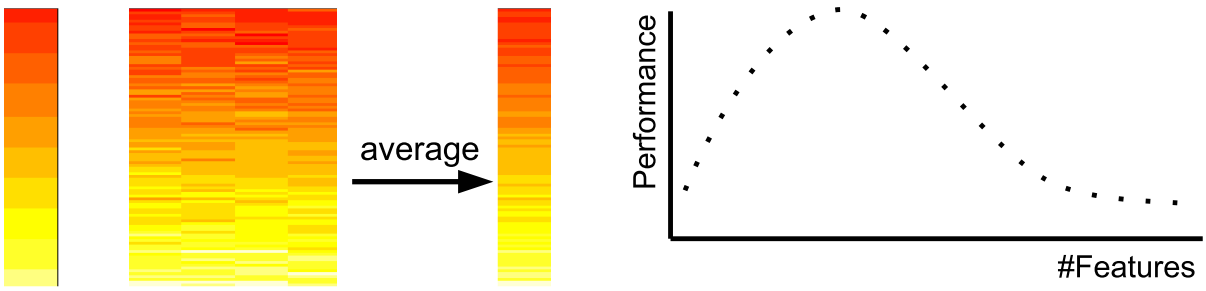
information to the model is added. There are two common approaches to determine which feature adds the most information: by wrapper and by filter. Wrappers train a classifier for each feature that can be added in each round. The resulting classification performance is then used to decide which feature should be selected in that round. Filters use a proxy measure instead of the performance of the model itself, e.g. the correlation to the residual. Since the computational complexity of wrappers is prohibitively high, we will use the filter approach.

When using correlation to the residual, for the first round, this would mean selecting the feature which has the maximum correlation to the output $\mathbf{y}$. In subsequent rounds, this would be the feature which is maximally correlated with the residual $y - f_{i-1}(\mathbf{X})$, where $f_{i-1}(\mathbf{X})$ is the output from the classifier trained using the features selected in previous rounds. In this work, we use logistic regression (as implemented in LiblineaR [16]) as a classifier f.

Forward Feature Selection results in a ranked list of features, indicating at which point they are added to the model. To determine which round corresponds to the optimal feature set, techniques such as cross-validation can be used (Figure 2c). The feature set corresponding to the best performance can then be used as input in a classifier for future predictions.

A detailed algorithm of Forward Feature Selection is provided below.

- Input: an $n \times p$ feature matrix $\mathbf{X}$ and an $n \times 1$ output vector $\mathbf{y}$.
- Let $\mathcal{F}_i$ be the set of selected features in round i and initialize $\mathcal{F}_0 = \{\}$
- Let $f_i(\mathbf{X})$ be the output from a classifier trained on the features in $\mathcal{F}_i$ and initialize $f_0(\mathbf{X}) = 0$.
- Let π be a $p \times 1$ vector, in which the performance of each $f_i(\mathbf{X})$ will be saved.
- For $i = 1 : p$
  - Compute the residual $\mathbf{y} - f_{i-1}(\mathbf{X})$.
  - Compute the correlation between each feature and the residual and store it in $\mathbf{r}$.
  - Let ν be the feature corresponding to the maximum in r.
  - Set $\mathcal{F}_i = \mathcal{F}_{i-1} \cup \nu$.
  - Train the classifier $f_i(\mathbf{X})$ using the features in $\mathcal{F}_i$.
  - Determine the performance of $f_i(\mathbf{X})$ and store it in $\pi[i]$
- Determine in which round the forward feature selection was optimal, i.e. which i corresponds to the maximum in $\pi[i]$.
- Output: the feature set $\mathcal{F}_i$



(a) (b) (c)

Figure 2: **a)** In regular forward feature selection, in every round the feature with the maximum correlation to the residual $\mathbf{y} - f_{i-1}(\mathbf{X})$ is selected. The vector depicted in panel (a) represents the correlations between each feature and the residual. **b)** In Multi-task Forward Feature Selection, we get such a vector per task, resulting in the matrix depicted in panel (b). In order to select the feature which has on average maximum correlation to the residuals of the tasks, we take the row-wise average of this matrix and select the feature corresponding to the maximum in the resulting vector. **c)** In order to determine in which round the set of selected features was optimal, we determine the performance for each number of features, resulting in the above graph. We can then select the set of features corresponding to the best performance.

3.2.2 Multi-task Forward Feature Selection

Multi-task Forward Feature Selection works in a similar fashion, but with one important difference: since Multi-task Learning considers multiple outputs simultaneously, the metric to determine which feature adds the most information needs to be redefined. We define this metric such that in each round the feature which has on average maximum correlation to the residuals of the different tasks is selected (Figure 2b). By averaging the correlations over the different tasks, we reduce the susceptibility to spurious correlations between a feature and a single residual.

In each round and for each task, a separate classifier $f_{ij}(\mathrm{X})$ is trained, where i indicates the round and j indicates the task. Like we did in Regular Forward Feature Selection, we use a logistic regression classifier as implemented in LiblineaR [16]. The algorithm for Multi-task Forward Feature Selection is provided below.

- Input: an $n \times p$ feature matrix $\mathbf{X}$ and an $n \times q$ output matrix $\mathbf{Y}$.
- Let $\mathcal{F}_i$ be the set of selected features in round i and initialize $\mathcal{F}_0 = \{\}$
- Let $f_{ij}(\mathbf{X})$ be the output from a model trained on the features in $\mathcal{F}_i$ for the output j. Initialize $f_{0j}(x) = 0$ for all $i \in [1:q]$.
- Let π be a $p \times q$ matrix, in which the performance of each $f_{ij}(\mathbf{X})$ will be saved.
- For $i = 1:p$
  - Let $\mathbf{R}$ be a $p \times q$ matrix in which the correlations between features and residuals will be stored.
  - For $j = 1:q$
    - Compute the residual for the jth task $y_j - f_{i-1,j}(\mathbf{X})$.
    - Compute the correlation between each feature and the residual and store it in the jth column of $\mathbf{R}$, i.e. in $\mathbf{R}_{*j}$.
  - Compute the row-wise average of $\mathbf{R}$, resulting in a $p \times 1$ vector $\mathbf{r}$ containing per feature the average correlation to each task.
  - Let ν be the feature corresponding to the maximum in $\mathbf{r}$.
  - Set $\mathcal{F}_i = \mathcal{F}_{i-1} \cup \nu$.
  - For $j = 1:q$

    - Train the model $f_{ij}(X)$ using the features in $\mathcal{F}_i$.
    - Determine the performance of $f_{ij}(X)$ and store it in π_{ij}
- Compute the row-wise average of π, resulting in a $p \times 1$ vector π_{avg} containing the average performance per iteration of feature selection.
- Determine in which round the forward feature selection was optimal, i.e. which i corresponds to the maximum in π_{avg}.
- Output: the feature set $\mathcal{F}_i$

3.3 Multi-task Learning using the Group Lasso and the Sparse Group Lasso

3.3.1 Lasso

Besides Forward Feature Selection, we will consider Lasso [30]: a feature selection heuristic based on L1-norm regularization. Consider for example L1-regularized logistic regression.

$$\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{n} \log\left(1 + e^{-y_i \mathbf{X}_{i*} \boldsymbol{\beta}^T}\right) + \lambda \left\|\boldsymbol{\beta}\right\|_1 \right] \tag{1}$$

Where $\mathbf{X}_{i*}$ is the feature vector of the ith sample and y_i is the output label of the ith sample. The first term is a logistic regression loss function, defining a $1 \times p$ mapping $\boldsymbol{\beta}$ that relates the p features in $\mathbf{X}$ to the corresponding values in $\mathbf{y}$. The second term is an L1-penalty, which enforces $\boldsymbol{\beta}$ to have only few non-zero values, ensuring the selection of just a small set of features. Using λ, a trade-off can be made between the how well $\boldsymbol{\beta}$ should be fit to the training data and the number of features that should be selected. In order to determine the optimal value of λ, techniques such as cross-validation can be used. Using Least Angle Regression [7] the entire regularization path can be computed simultaneously, i.e. the solutions for all possible λ can be determined in one go, which significantly speeds up the cross-validation.

3.3.2 Group Lasso and Sparse Group Lasso

The Group Lasso [32] is a variation of Lasso in which groups of features are regularized together, such that sparsity is enforced at group level instead of feature level. This is achieved by nesting an L2-norm within an L1-norm in the following way. For each group of features $g \in G$, the corresponding group of coefficients $\boldsymbol{\beta}_g \subseteq \boldsymbol{\beta}$ is penalized using an L2-norm: $\left\|\boldsymbol{\beta}_g\right\|_2$. The resulting L2-norms are then inserted into an L1-norm:

$$\begin{aligned} &\left\| \left\|\boldsymbol{\beta}_g\right\|_2, \forall g \in G \right\|_1 \\ =& \sum_{g \in G} \left| \left\|\boldsymbol{\beta}_g\right\|_2 \right| \\ =& \sum_{g \in G} \left\|\boldsymbol{\beta}_g\right\|_2 \end{aligned} \tag{2}$$

Due to the L1-norm, only a few groups $\boldsymbol{\beta}_g$ can be non-zero. For each group of non-zero $\boldsymbol{\beta}_g$, the L2-norm will enforce that all $\boldsymbol{\beta} \in \boldsymbol{\beta}_g$ will be non-zero, resulting in the selection of complete groups of features.

We will consider the Group Lasso constraint in a logistic regression.

$$\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{n} \log\left(1 + e^{-y_i \mathbf{X}_{i*} \boldsymbol{\beta}^T}\right) + \lambda \sum_{g \in G} \sqrt{p_g} \left\|\boldsymbol{\beta}_g\right\|_2 \right] \tag{3}$$

Where the $\sqrt{p_g}$ accounts for varying group sizes. In order to reduce notational complexity and because we will only consider groups of equal size, we have omitted the weights $\sqrt{p_g}$ in all further equations. In Section 3.3.3 we will discuss how the groups $g \in G$ can be defined for application in Multi-task Learning.

A variation on the Group Lasso is the Sparse Group Lasso [10, 26], which combines L1-regularization and the Group Lasso. While the Group Lasso will enforce the selection of complete groups of features,

the addition of L1-regularization allows for sparsity within these groups. This enables the method to select part of the features in a group, instead of strictly forcing the entire group to be selected. Once again, we will consider this constraint in a logistic regression.

$$\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{n} \log\left(1 + e^{-y_i \mathbf{X}_{i*} \boldsymbol{\beta}^T}\right) + \lambda_1 \sum_{g \in G} \left\|\boldsymbol{\beta}_g\right\|_2 + \lambda_2 \left\|\boldsymbol{\beta}\right\|_1 \right] \tag{4}$$

The optimal values of λ_1 and λ_2 can be determined using cross-validation. For both the Group Lasso and the Sparse Group Lasso, there also exist algorithms that compute the entire regularization path, allowing faster cross-validation. In all experiments involving these constraints, we have used the R package SGL [25], which implements such an algorithm.

3.3.3 Multi-task Group Lasso and Multi-task Sparse Group Lasso

In order to use the Group Lasso or the Sparse Group Lasso for Multi-task Learning, we will define a joint loss function, in which we optimize over all tasks simultaneously, and we will define the groups of features $g \in G$ such that the feature selection will be performed jointly over all tasks (Figure 3).

Consider a $n \times q$ matrix $\mathbf{Y}$ as the concatenation of the output vectors from q tasks. Given a $n \times p$ input matrix $\mathbf{X}$, we want to find a $q \times p$ matrix $\boldsymbol{\beta}$ such that:

$$\mathbf{Y} = \frac{1}{1 + e^{-\mathbf{X}\boldsymbol{\beta}^T}} = f\left(\mathbf{X}\boldsymbol{\beta}^T\right) \tag{5}$$

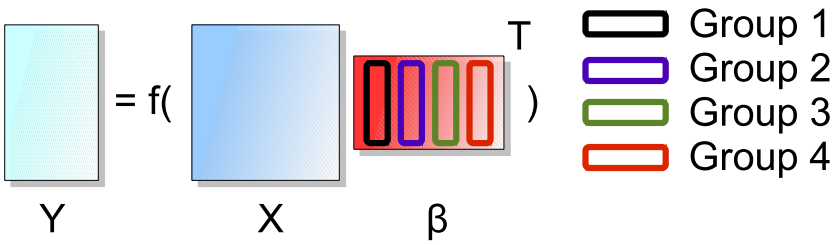


Figure 3: A schematic representation of Multi-task Learning Group Lasso. Consider an input matrix X with 4 features. In order to tie together the selection of features over the various task, we construct 4 groups: one per feature. Each feature in each task is then added to its corresponding group.

To derive this $\boldsymbol{\beta}$ from $\mathbf{X}$ and $\mathbf{Y}$, we define the following joint loss-function:

$$\min_{\boldsymbol{\beta}} \sum_{j=1}^{q} \sum_{i=1}^{n} \log\left(1 + e^{-y_{ij} \mathbf{X}_{i*} \boldsymbol{\beta}_{j*}^T}\right) \tag{6}$$

Where q is the number of tasks, y_{ij} is the output for the ith sample of the jth task, $\mathbf{X}_{i*}$ is the feature vector for the ith sample and $\boldsymbol{\beta}_{j*}$ are the regression coefficients for the jth task. In order to solve this problem using existing algorithms, we rewrite this to a regular logistic regression problem by defining a augmented input matrix $\mathbf{X}^*$ as

$$\mathbf{X}^* = \underbrace{\begin{bmatrix} \mathbf{X} & 0 & \dots & 0 \\ 0 & \mathbf{X} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mathbf{X} \end{bmatrix}}_{q \text{ times}} \tag{7}$$

And an augmented output vector $\mathbf{y}^*$ as

$$\mathbf{y}^* = \begin{bmatrix} \mathbf{Y}_{*1} \\ \mathbf{Y}_{*2} \\ \vdots \\ \mathbf{Y}_{*q} \end{bmatrix} \tag{8}$$

Where each $\mathbf{Y}_{*j}$ is the output vector for the jth task. Finally, we define an augmented coefficient vector $\boldsymbol{\beta}^*$ as

$$\boldsymbol{\beta}^* = \begin{bmatrix} \boldsymbol{\beta}_{1*} & \boldsymbol{\beta}_{2*} & \dots & \boldsymbol{\beta}_{q*} \end{bmatrix} \tag{9}$$

Where each $\boldsymbol{\beta}_{j*}$ contains the regression coefficient for the jth task. When we insert $\mathbf{X}^*$, $\mathbf{y}^*$ and $\boldsymbol{\beta}^*$ in a logistic regression loss, we can see that it is equal to equation 6.

$$\min_{\boldsymbol{\beta}^*} \sum_{i=1}^{nq} \log\left(1 + e^{-y_i^* \mathbf{X}_i^* \boldsymbol{\beta}^{*T}}\right) = \min_{\boldsymbol{\beta}} \sum_{j=1}^{q} \sum_{i=1}^{n} \log\left(1 + e^{-y_{ij} \mathbf{X}_{i*} \boldsymbol{\beta}_{j*}^T}\right) \tag{10}$$

However, note that the above optimization does not differ from fitting the coefficients for each task individually. In order to tie together the different tasks, we will use the Group Lasso. For each feature in $\mathbf{X}$, we define a group, resulting in p groups: $G = [g_1 ... g_p]$. Then for each task, we add feature 1 to group 1, feature 2 to group 2, etc. (Figure 4). This results in the following Group Lasso optimization:

$$\begin{aligned} &\min_{\boldsymbol{\beta}^*} \left[\sum_{i=1}^{nq} \log\left(1 + e^{-y_i^* \mathbf{X}_{i*}^* \boldsymbol{\beta}^{*T}}\right) + \lambda \sum_{g \in G} \left\| \boldsymbol{\beta}_g \right\|_2 \right] \\ =&\min_{\boldsymbol{\beta}} \left[\sum_{j=1}^{q} \sum_{i=1}^{n} \log\left(1 + e^{-y_{ij} \mathbf{X}_{i*} \boldsymbol{\beta}_{j*}^T}\right) + \lambda \sum_{g \in G} \left\| \boldsymbol{\beta}_g \right\|_2 \right] \end{aligned} \tag{11}$$

As an intuitive interpretation of the optimization above, observe that when it seems beneficial to select a certain feature for one task, the Group Lasso constraint enforces also selecting the same feature for other tasks. This results in simultaneous feature selection over the different tasks.

We extend this form of Multi-task Learning by using the Sparse Group Lasso.

$$\begin{aligned} &\min_{\boldsymbol{\beta}^*} \left[\sum_{i=1}^{nq} \log\left(1 + e^{-y_i^* \mathbf{X}_{i*}^* \boldsymbol{\beta}^{*T}}\right) + \lambda_1 \sum_{g \in G} \left\| \boldsymbol{\beta}_g \right\|_2 + \lambda_2 \left\| \boldsymbol{\beta} \right\|_1 \right] \\ =&\min_{\boldsymbol{\beta}} \left[\sum_{j=1}^{q} \sum_{i=1}^{n} \log\left(1 + e^{-y_{ij} \mathbf{X}_{i*} \boldsymbol{\beta}_{j*}^T}\right) + \lambda_1 \sum_{g \in G} \left\| \boldsymbol{\beta}_g \right\|_2 + \lambda_2 \left\| \boldsymbol{\beta} \right\|_1 \right] \end{aligned} \tag{12}$$

The Sparse Group Lasso constraint allows selecting a feature for just part of the tasks. Even though we expect similar mechanisms underlying related drugs, there may be features which are specific for just certain tasks. Using the Sparse Group Lasso allows us to deal with such situations, at the cost of increased complexity.

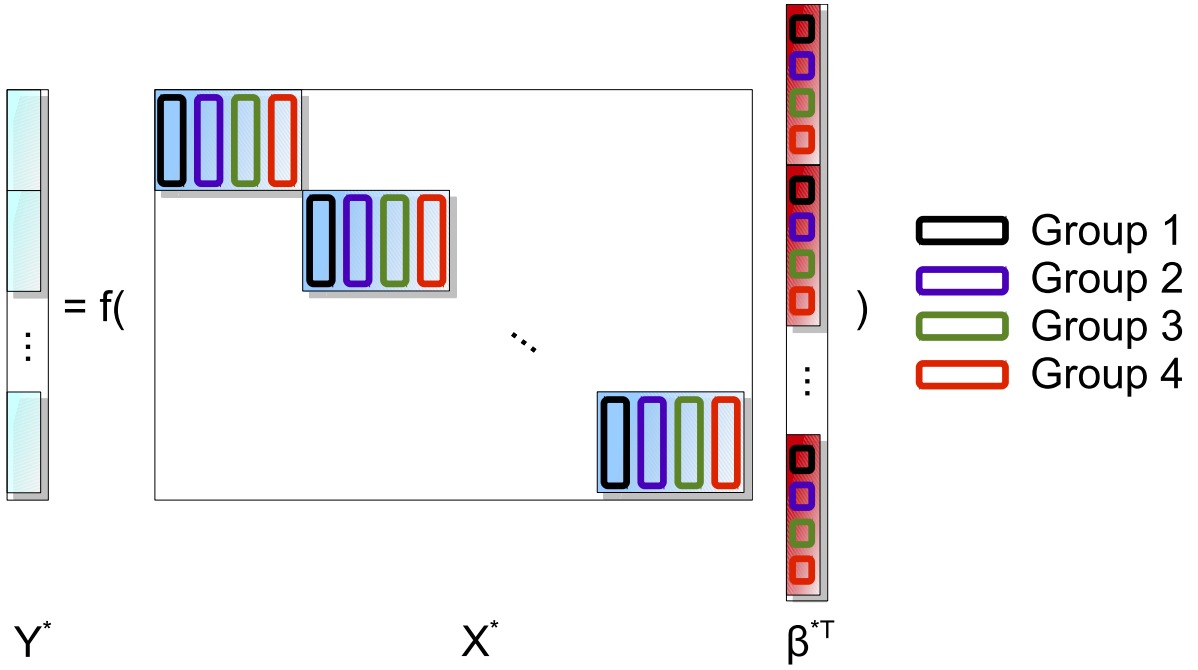


Figure 4: A schematic representation of the augmented representation of Multi-task Learning Group Lasso using $\mathbf{X}^*$, $\mathbf{Y}^*$ and $\boldsymbol{\beta}^*$. Once again, we show the positioning of the 4 group-constraints.

4 Results

4.1 Artificial data experiments

In order to test the performance of Multi-task Learning in a situation where the ground truth is known, we have constructed an artificial data set with properties similar to those of the cell line panel. Using this data set, we compare various Multi-task Learning techniques, as well as Single-task Learning.

4.1.1 Construction of the data

We generated an input matrix $\mathbf{X}$ using a standard-normal random generator, with $n = 500$ samples and $p = 13.000$ features, similar to the size of the drug sensitivity data set. In order to simulate the multivariate character often observed in biological data, we generated an output vector $\mathbf{y}$ as a combination of the first three features in $\mathbf{X}$: $\mathbf{y}_{\text{real}} = (\mathbf{X}_{*1} + \mathbf{X}_{*2} + \mathbf{X}_{*3}) > t$. We use a threshold t in order to obtain a binary output vector, where the value of t is chosen such that the class-balance is 9:1, similar to the ratio of sensitive vs. resistant cell lines. Of course, if we were to train a feature selection model on $\mathbf{X}$ and $\mathbf{y}_{\text{real}}$, we would expect it to select feature 1, 2 and 3.

Since the clusters of drugs in the cell line panel are usually of two or four, we will turn this into a Multi-task Learning data set with $q = 3$ different tasks. To this end, we replicate $\mathbf{y}_{\text{real}}$ three times and store these replicates in the $n \times q$ matrix $\mathbf{Y}$, such that each column is identical to the original output vector $\mathbf{y}_{\text{real}}$. Then, independently for each task, we introduce noise by randomly swapping $k\%$ of the labels, where k is set to 20 unless otherwise mentioned. This results in 3 outputs which are different from each other, yet still correlated with each other and with the input variables.

We also consider two variations of the above data set. In the first variation, the output of the third task is defined as $\mathbf{y}'_{\text{real}} = (\mathbf{X}_{*1} + \mathbf{X}_{*2} + \mathbf{X}_{*4}) > t$, thus introducing a task which depends on a slightly different set of features. In the second variation, we ensure one of the tasks is completely unrelated by defining the output of the third task as $\mathbf{y}''_{\text{real}} = (\mathbf{X}_{*5} + \mathbf{X}_{*6} + \mathbf{X}_{*7}) > t$.

4.1.2 Experimental set-up

We will discuss five different methods described in the Methods section, which are summarized in table 1. In order to reduce the running-time of the experiments, we limit the total number of features that can be selected to 100. For the Forward Feature Selection methods, this means that after 100 features are added, the method terminates. For the Multi-task Learning based on the (Sparse) Group Lasso, for each cross-validation fold we perform a pre-selection in which the 100 features with the maximum average correlation to the outputs are selected.

Method	Description
Regular Forward Feature Selection	The Single-task version of forward feature selection
Multi-task Forward Feature Selection	The Multi-task version of forward feature selection, which averages the correlations to the residual over tasks in each round
Majority Vote Meta-task	A simple Multi-task Learning algorithm, which creates a meta-task by taking the majority vote of the output labels and uses regular, Single-task forward feature selection on the meta-task
Single-task Lasso	A Single-task Learning algorithm using L1-regularization
Multi-task Group Lasso	Multi-task Learning using Group Lasso constraints
Multi-task Sparse Group Lasso	Multi-task Learning using a combination of Group Lasso and L1-regularization constraints

Table 1: Methods used in artificial data set experiments.

For each experiment, we will report three performance measures. First, we will report the AUC, which we compute using $\mathbf{y}_{\text{real}}$. Next, we will compare the set of selected features to the set of features used to define the output of a task. From this, we will report the feature TPs, i.e. the number of correctly selected features, and the feature FNs, i.e. the number of features that are incorrectly selected. These numbers can be determined per task. However, we will report the average of these numbers over the tasks. Finally, for each experiment we repeated the construction and analysis of the data 10 times. From this we compute the mean and standard deviation, which in the figures are reported as error bars.

4.1.3 Sparse Group Lasso solver yields poor performance in the Lasso-based methods

In Figure 5 we compare the classification performance of all methods at a label noise level of $k = 10\%$. Unfortunately, the performance for Multi-task Sparse Group Lasso is missing, because the R-package SGL, which we used to optimize the corresponding Sparse Group Lasso problem, consistently crashed.

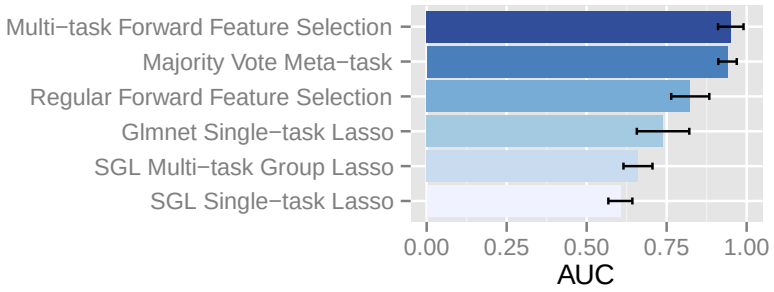


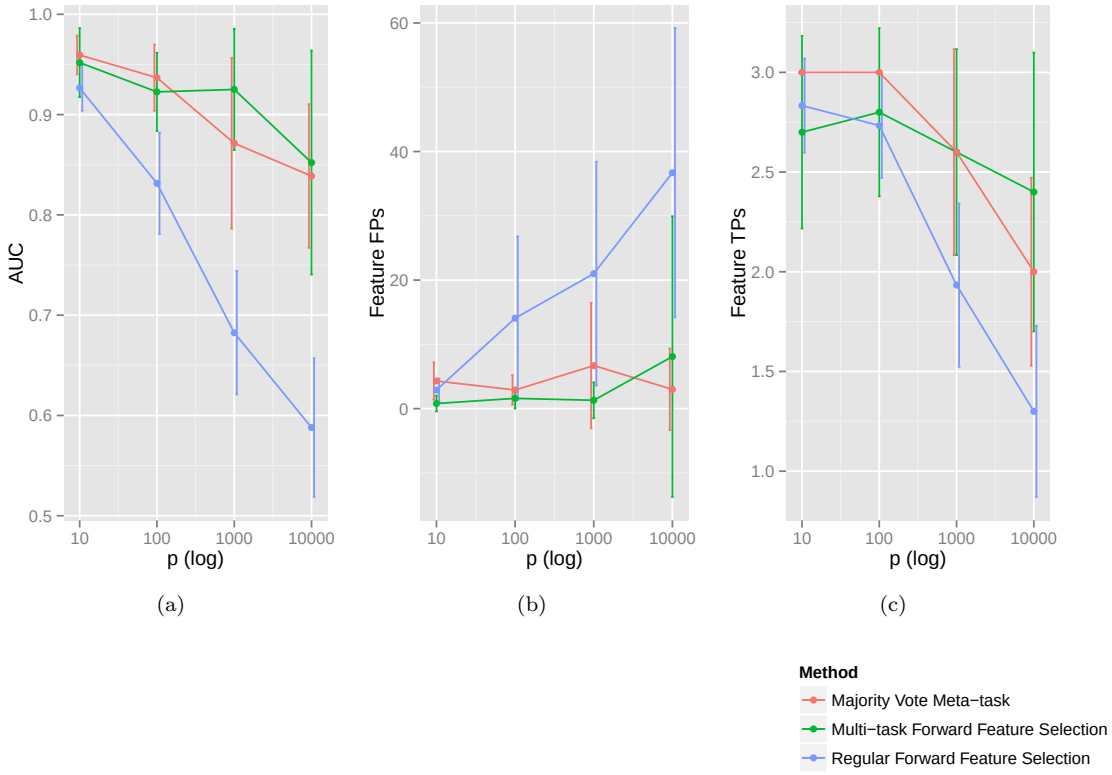
Figure 5: Classification performance of all methods at a label noise level of 10%. The methods for which SGL was used as a solver perform worst. In the case of Single-task Lasso, replacing SGL with Glmnet resulted in a significant gain in performance. This made us question the SGL package.

The same package was used on the same data to solve the optimization of the Multi-task Group Lasso and Single-task Lasso. Surprisingly, the resulting classification performance was lower than all other methods. This made us question the SGL package, since we would expect these methods to have performance similar to the methods based on Forward Feature Selection. To rule out an error in our code, we repeated the Single-task Lasso experiment using Glmnet, a widely used Lasso-solver. Indeed, this did result in performance similar to Regular Forward Feature Selection, ruling out a mistake in our implementation.

We suggest that in future work different Group Lasso solvers should be used to test these algorithms, e.g. SLEP [18]. Unfortunately, in this project there was not enough time left to do such experiments. Therefore, in the rest of the experiments discussed, we will not consider the Lasso-based methods anymore and instead we will focus on the methods based on Forward Feature Selection.

4.1.4 Multi-task Learning outperforms Single-task Learning in an undersampled situation

In the second experiment, we investigated how Multi-task Learning performs in increasingly undersampled settings. To this end, we varied the dimensionality of the artificial data set, while keeping all other parameters fixed. In Figure 6a, we see that the performance of Multi-task Forward Feature Selection and Majority Vote Meta-task remain above that of the Regular Forward Feature Selection for all settings of the dimensionality. The reason for this large difference in performance is partly explained by the feature FPs (Figure 6b): the Multi-task Learning methods barely select any wrong features. On the other end, Regular Forward Feature Selection selects increasingly more incorrect features as the dimensionality is increased. Finally, in Figure 6c we see that the Multi-task Learning methods also perform slightly better on the feature TPs. We reason that Multi-task Learning exploits the similarity between the different tasks, leading to a better selection of features, which in turn results in a better classification performance.



(a) (b) (c)

Figure 6: Comparison of the influence of the dimensionality, based on: **a)** the classification performance; **b)** the feature FPs: the number of features that are incorrectly selected; and **c)** the feature TPs: the number of features that is correctly selected.

We repeated this experiment with the first variation of this data set, in which a slightly different task is included. The results are very similar, but there is one important difference: Majority Vote Meta-task performs much better than Multi-task Forward Feature Selection (Figure 7a). In the second variation, in which a completely unrelated task is included, Majority Vote Meta-task also achieves the highest classification performance (Figure 7d). However, in terms of feature selection, Multi-task Forward Feature Selection performs better (Figure 7e and 7f). In these experiments we see that even though a task is included which lacks similarity to the other tasks, Multi-task Learning can still be beneficial.

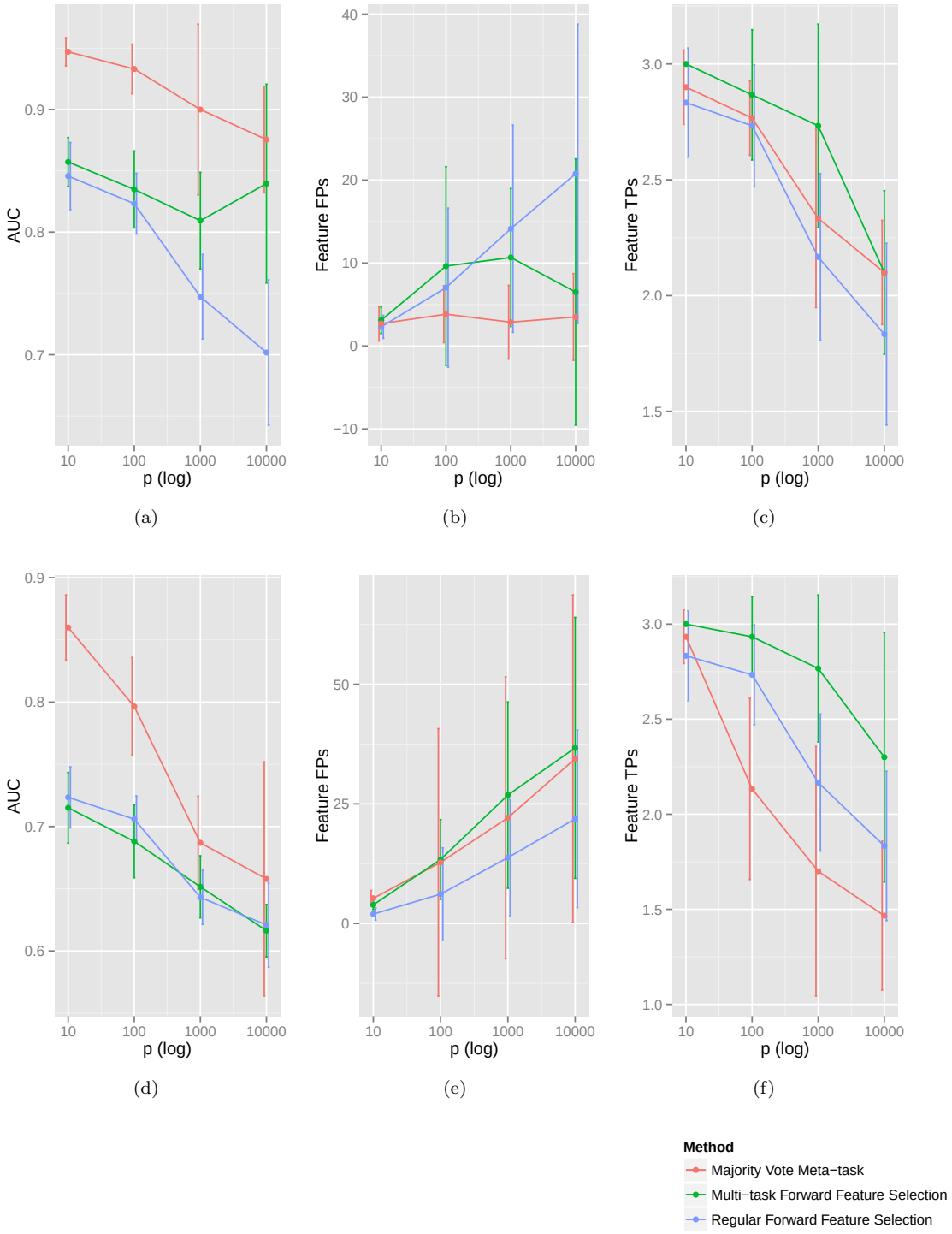


Figure 7: Comparison of the influence of the dimensionality. For the first variation, in which a slightly different task is included, we show: **a)** the classification performance; **b)** the feature FPs; and **c)** the feature TPs. Similarly, for the second variation, in which a completely unrelated task is included, we show: **d)** the classification performance; **e)** the feature FPs; and **f)** the feature TPs.

4.1.5 Multi-task Learning successfully exploits similarity between tasks

In this experiment we analyzed how the amount of label noise in each task influences Multi-task Learning, by varying k (the percentage of labels that are swapped per task), again keeping all other parameters fixed. In Figure 8a, we see that the two Multi-task Learning methods, namely Multi-task Forward Feature Selection and Majority Vote Meta-task, consistently outperform Regular Forward Feature Selection. Also when considering the feature FPs (Figure 8b) and the Feature TPs (Figure 8c), both Multi-task Learning methods perform slightly better than the Single-task Learning approach. In this experiment, Multi-task Learning shows it is able to 'average out' the label noise by considering multiple tasks simultaneously.

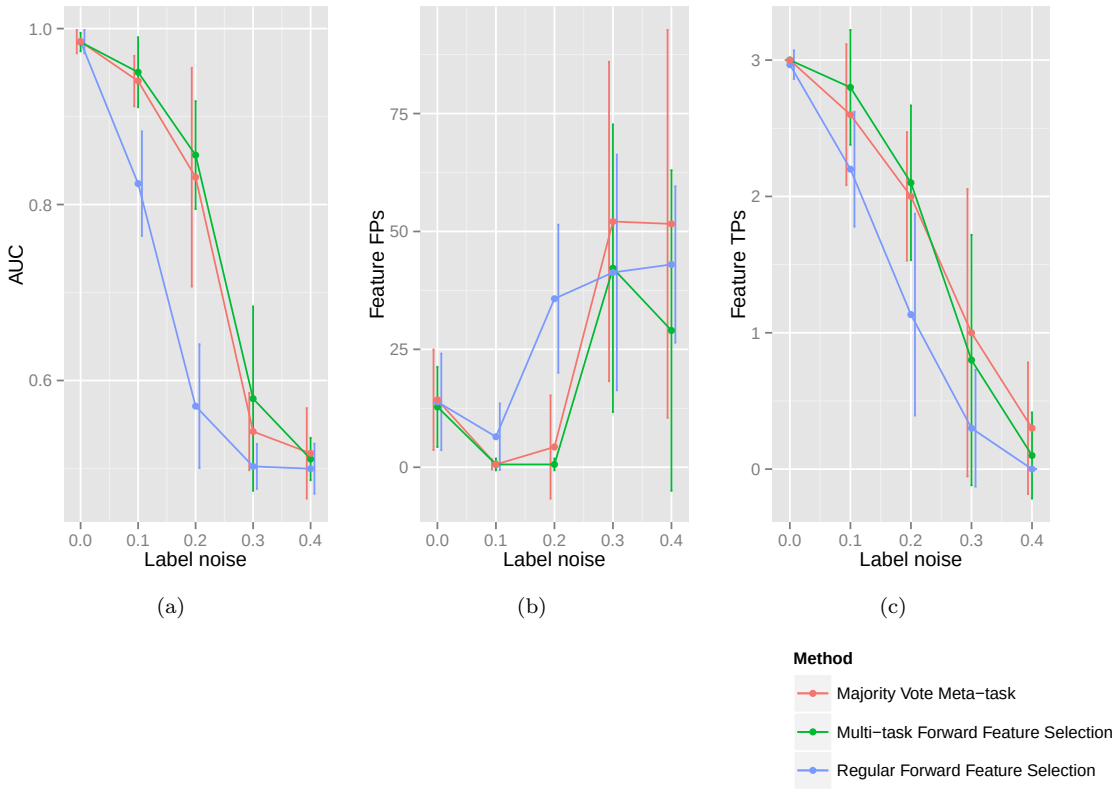


Figure 8: Comparison of the influence of label noise between the various tasks, based on: **a)** the classification performance; **b)** the feature FPs: the number of features that are incorrectly selected; and **c)** the feature TPs: the number of features that is correctly selected.

When repeating this experiment with the first variation of our data set, we obtain very similar results (Figure 9a). For the second variation of the data set, the performance of the different methods lies very close to each other, though Majority Vote Meta-task does achieve slightly better performance than the rest (Figure 9d). Again, we see here that even when a (slightly) unrelated task is included, Multi-task Learning can be beneficial.

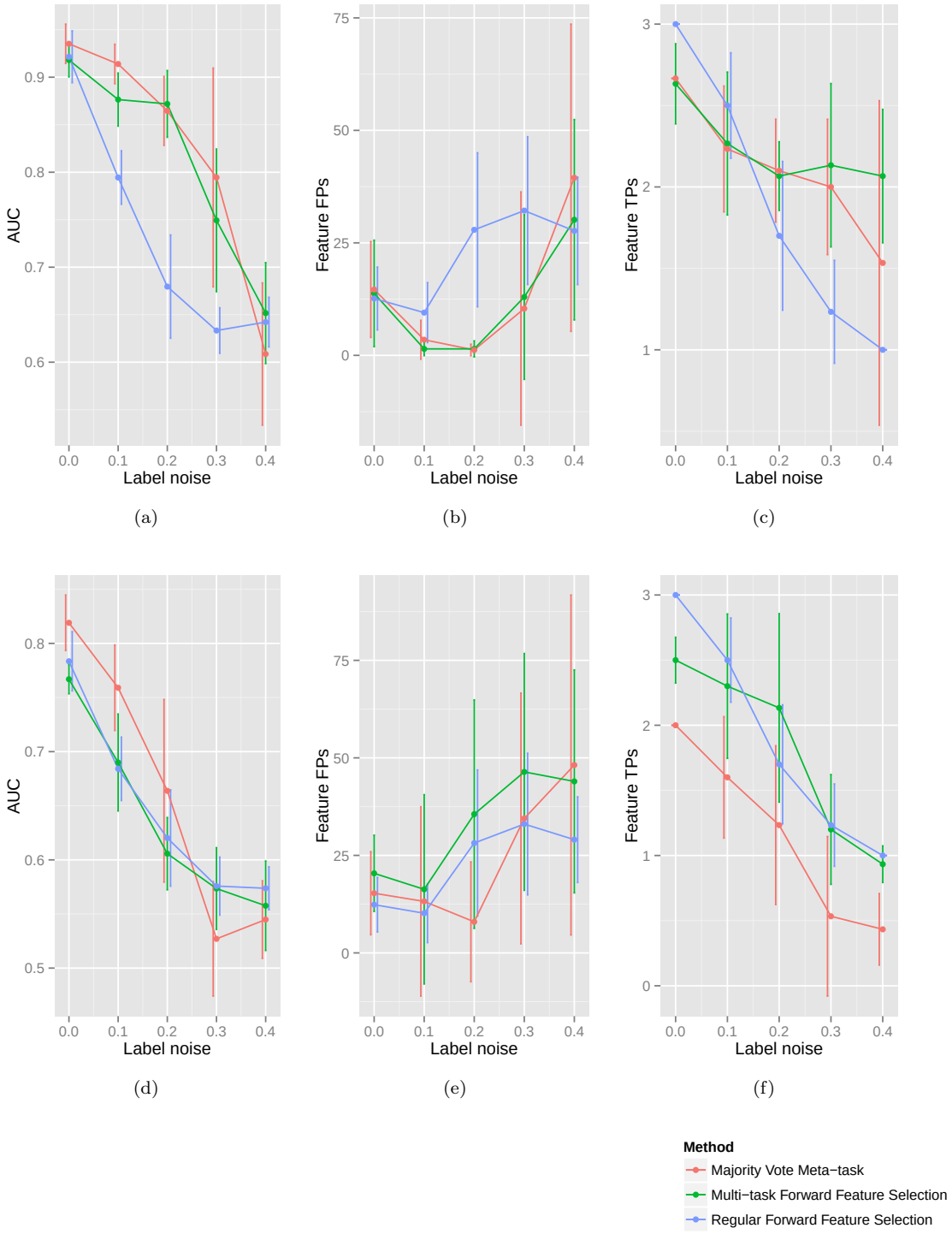


Figure 9: Comparison of the influence of label noise between the various tasks. For the first variation, in which a slightly different task is included, we show: **a)** the classification performance; **b)** the feature FPs; and **c)** the feature TPs. Similarly, for the second variation, in which a completely unrelated task is included, we show: **d)** the classification performance; **e)** the feature FPs; and **f)** the feature TPs.

4.2 Application to the drug sensitivity screen

4.2.1 Identification of groups of similar drugs

In order to identify groups of similar drugs, we performed cluster analysis on the drug response over the cell lines in the screen. Prior to the clustering, we selected the drugs for which Regular Forward Feature Selection resulted in at least an AUC of 0.7. Next, we have used hierarchical clustering as implemented in R [23]. To determine the distance between all drugs, we used Pearson correlation. As linkage criterion we have chosen complete linkage, in order to obtain small, compact clusters. In the dendrogram we set a cut-off at both 1.5 and at 1.7, resulting in 5 and 9 clusters respectively. In Figure 10, the clustering is shown. For a more detailed overview including a heatmap, we refer to Supplementary Figure S1.

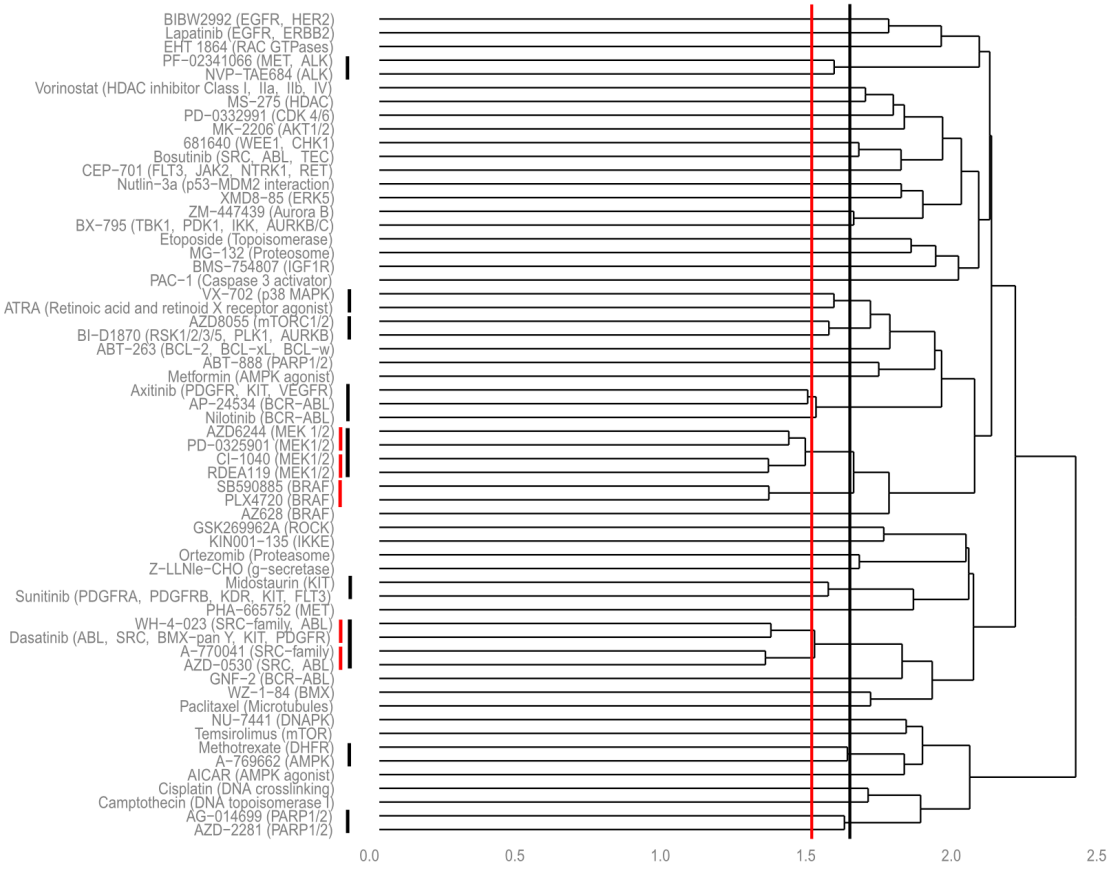


Figure 10: Hierarchical clustering of the drugs in the cell line panel data. The black line and the red line indicate the different cut-offs used for the dendrogram. The resulting clusters are indicated in black and red respectively.

Cluster	Drug name	Drug target
1	PLX4720	BRAF
1	SB590885	BRAF
2	AZD-0530	SRC, ABL
2	A-770041	SRC-family
3	Dasatanib	SRC-family, ABL
3	WH-4-023	SRC, ABL, BMX-pan Y, KIT, PDGFR
4	RDEA119	MEK1/2
4	CI-1040	MEK1/2
5	PD-0325901	MEK1/2
5	AZD6244	MEK1/2
6	RDEA119	MEK1/2
6	CI-1040	MEK1/2
6	PD-0325901	MEK1/2
6	AZD6244	MEK1/2
7	AZD-0530	SRC, ABL
7	A-770041	SRC-family
7	Dasatanib	SRC-family, ABL
7	WH-4-023	SRC, ABL, BMX-pan Y, KIT, PDGFR
8	AZD-2281	PARP1/2
8	AG-014699	PARP1/2
9	A-769662	DHFR
9	Methotrexate	AMPK
10	Sunitinib	KIT, PDGFRA, PDGFRB, KDR, FLT3
10	Midostaurin	KIT
11	AP-24534	BCR-ABL
11	Nilotinib	BCR-ABL
11	Axitinib	AMPK agonist
12	NVP-TAE684	ALK
12	PF-02341066	ALK, MET
13	VX-702	p38 MAPK
13	ATRA	Retinoic acid and retinoid X recepter agonist
14	AZD8055	mTORC1/2
14	BI-D1870	RSK1/2/3/5, PLK1, AURKB

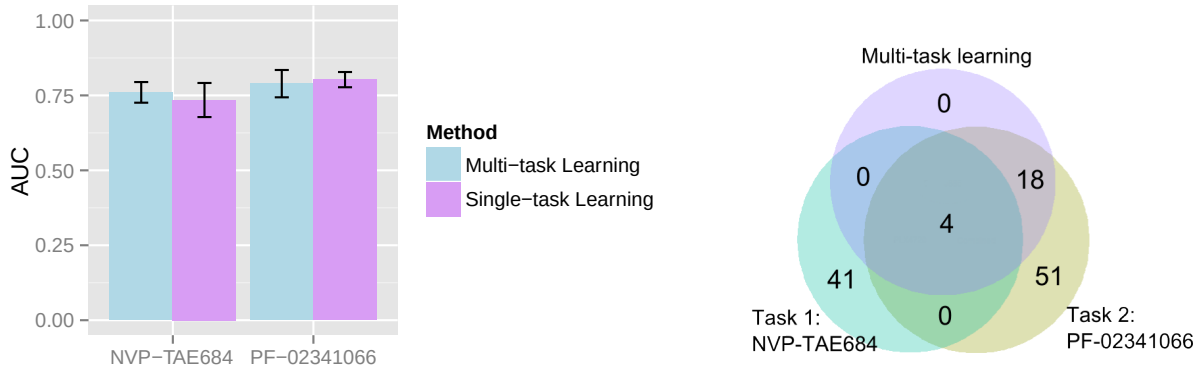
Table 2: Clusters identified by the cluster analysis.

Most of the identified clusters share a common drug target, though some notable exceptions are clusters 9, 13, 14 and to some extent cluster 11. We did not further investigate what causes these drugs to cluster together.

In the following experiments, we have used all of the above clusters to define groups of drugs on which we perform Multi-task Learning. As most of the clusters are of size two, using Majority Vote Meta-task would be problematic: for every cell line that is sensitive to one drug in the cluster, but resistant to the other, it is unclear how the meta-task should be defined. Since Multi-task Forward Feature Selection achieves similar performance in the artificial data set experiments, we will focus on this method.

4.2.2 Comparing Multi-task Learning and Single-task Learning

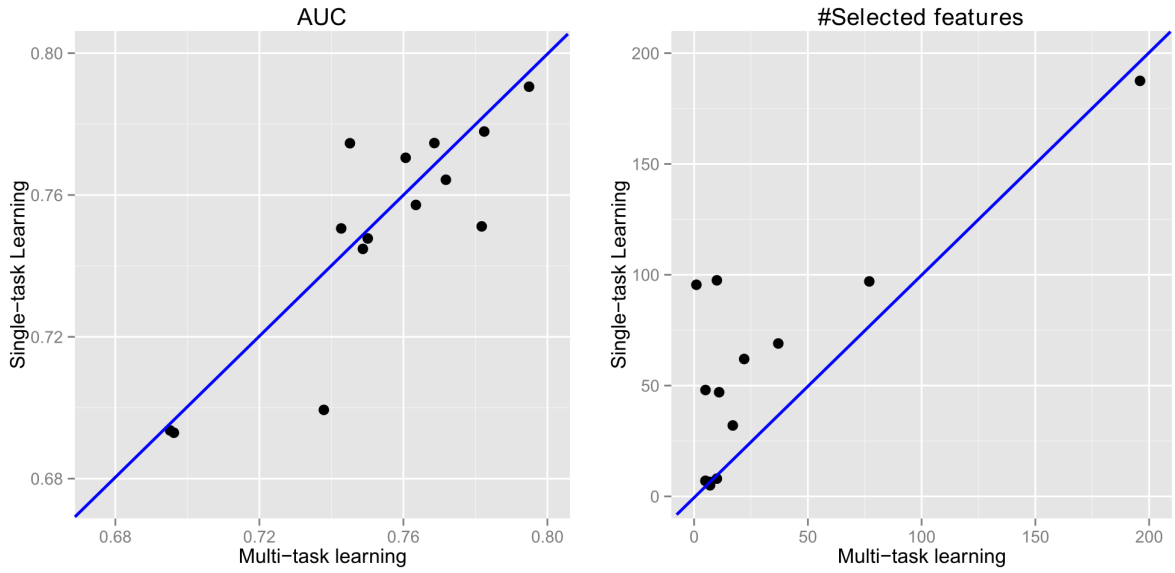
In Figure 11a, both the Single-task Learning and the Multi-task Learning classification performance is shown for two ALK inhibitors: NV-TAE684 and PF-02341066. From this barplot, it can be seen that Multi-task Learning and Single-task Learning result in similar performance. However, when we consider the selected features (Figure 11b), we observe that Multi-task Learning is able to obtain this performance using a smaller set of features. While Single-task Learning selects 45 features for NVP-TAE684 and 55 features for PF-02341066, the Multi-task Learning version only selects 22 features. We argue that since this smaller set of features can equally well explain the drug sensitivity, it might be a better set of features. This corresponds to the observations in artificial data set experiments, where Multi-task Learning resulted in fewer spuriously selected features compared to Single-task Learning.



(a) (b)

Figure 11: Comparison of Multi-task Learning and single-task learning on the ALK inhibitors NVP-TAE684 and PF-02341066, based on **a)** the classification performance and **b)** a Venn diagram of the selected features.

When applying Multi-task Learning to the other clusters of drugs, we observe similar results. Figure 12a shows that the classification performance of Multi-task Learning and Single-task Learning is very similar for each of these clusters. In Figure 12b we observe that the number of selected features is on average lower for Multi-task Learning than for Single-task Learning.



(a) (b)

Figure 12: Comparison of Multi-task Learning and Single-task Learning on the various clusters identified in the panel, using: **a)** the classification performance; and **b)** the number of features selected

Even though Multi-task Learning is able to achieve similar performance using a smaller set of features, it would be interesting to say something about the biological relevance of the features. For future work, we propose that the selected features are further inspected. For example, it would be interesting to see if the features selected by Multi-task Learning are more often annotated with certain pathways, specially those associated with the drug under investigation.

5 Conclusion

In the analysis of drug sensitivity, we are faced with a severely undersampled problem, making statistical analysis challenging. Due to the undersampled nature of the cell line panel, many features have a correlation with the drug response that equals that of the features which actually cause the drug sensitivity. This results in spuriously selected features, which in turn confounds the identification of genomic markers in follow-up experiments.

We have shown that Multi-task Learning can be a useful tool to tackle the problem of spuriously selected features. By performing the feature selection simultaneously for multiple drugs, Multi-task Learning is less susceptible to spurious correlations between the genomic features and the drug response. We have discussed one existing Multi-task Learning technique as well as three novel ones, based on Group Lasso, Sparse Group Lasso and Forward Feature Selection respectively.

Using artificial data experiments, we have shown that Multi-task Learning leads to higher classification performance and a more accurate set of selected features when combining similar tasks. In two experiments, we varied the amount of label noise between tasks and the dimensionality respectively. In both cases, the classification performance of Multi-task Learning remained ahead of Single-task Learning. When varying the dimensionality, Multi-task Learning also resulted in far fewer spuriously selected features.

In the drug sensitivity screen, we determined 14 groups of drugs using clustering. For these 14 groups, we observe that Multi-task Learning selects far fewer features, while retaining similar performance. Since the smaller set of features explains the drug sensitivity equally well, we argue that it could be a better set of features. This corresponds to our observations in the artificial data set experiments, where Multi-task Learning reduces the number of spuriously selected features.

5.1 Future work

We see many opportunities to improve upon this concept in future work. First, we suggest to compare the sets of selected features by performing 'enrichment tests', to check if the selected features have a relation to known pathways associated with the drug under investigation. This way, the biological relevance of features selected using Multi-task Learning and Single-task Learning could be quantitatively compared.

Second, we suggest to further investigate the way drugs are clustered. For example, it might be interesting to apply Multi-task Learning to larger groups of drugs, even if the drugs in these groups will be less similar to each other. Alternatively, instead of clustering based on the drug response over the cell line panel, it could be interesting to perform clustering based on chemical properties of the drugs.

Next, we have discussed Multi-task Learning methods based on the (Sparse) Group Lasso, but have been unable to show their potential, possibly due to an error in the solver used. Therefore, we suggest that in future work different solvers, such as SLEP [18], are used to test these algorithms.

Finally, we think it could be interesting to use Multi-task Learning to combine the drug sensitivity screens of Barretina et al. [2] and Garnett et al. [12]. As noted by Haibe-Kains et al. [14], the genomic measurements in these screens are very similar, but there seem to be discrepancies in the drug sensitivity data. Multi-task Learning might be used here as a means to handle the possible label noise.

References

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[2] Jordi Barretina, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A Margolin, Sungjoon Kim, Christopher J Wilson, Joseph Lehár, Gregory V Kryukov, Dmitriy Sonkin, et al. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603–607, 2012.

[3] Giordano Caponigro and William R Sellers. Advances in the preclinical testing of cancer therapeutic hypotheses. *Nature Reviews Drug Discovery*, 10(3):179–187, 2011.

[4] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[5] Paul B Chapman, Axel Hauschild, Caroline Robert, John B Haanen, Paolo Ascierto, James Larkin, Reinhard Dummer, Claus Garbe, Alessandro Testori, Michele Maio, et al. Improved survival with vemurafenib in melanoma with braf v600e mutation. *New England Journal of Medicine*, 364(26):2507–2516, 2011.

[6] Jonathan R Dry, Sandra Pavey, Christine A Pratilas, Chris Harbron, Sarah Runswick, Darren Hodgson, Christine Chresta, Rose McCormack, Natalie Byrne, Mark Cockerill, et al. Transcriptional pathway signatures predict mek addiction and response to selumetinib (azd6244). *Cancer research*, 70(6):2264–2273, 2010.

[7] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[8] A Evgeniou and Massimiliano Pontil. Multi-task feature learning. In *Advances in neural information processing systems: Proceedings of the 2006 conference*, volume 19, page 41. The MIT Press, 2007.

[9] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[10] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

[11] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[12] Mathew J Garnett, Elena J Edelman, Sonja J Heidorn, Chris D Greenman, Anahita Dastur, King Wai Lau, Patricia Greninger, I Richard Thompson, Xi Luo, Jorge Soares, et al. Systematic identification of genomic markers of drug sensitivity in cancer cells. *Nature*, 483(7391):570–575, 2012.

[13] Levi A Garraway, Hans R Widlund, Mark A Rubin, Gad Getz, Aaron J Berger, Sridhar Ramaswamy, Rameen Beroukhim, Danny A Milner, Scott R Granter, Jinyan Du, et al. Integrative genomic analyses identify mitf as a lineage survival oncogene amplified in malignant melanoma. *Nature*, 436(7047):117–122, 2005.

[14] Benjamin Haibe-Kains, Nehme El-Hachem, Nicolai Juul Birkbak, Andrew C Jin, Andrew H Beck, Hugo JWL Aerts, and John Quackenbush. Inconsistency in large pharmacogenomic studies. *Nature*, 2013.

[15] Douglas Hanahan and Robert A Weinberg. The hallmarks of cancer. *cell*, 100(1):57–70, 2000.

[16] Thibault Helleputte. *LiblineaR: Linear Predictive Models Based On The Liblinear C/C++ Library*, 2013. R package version 1.80-7.

[17] Eunice L Kwak, Yung-Jue Bang, D Ross Camidge, Alice T Shaw, Benjamin Solomon, Robert G Maki, Sai-Hong I Ou, Bruce J Dezube, Pasi A Jänne, Daniel B Costa, et al. Anaplastic lymphoma kinase inhibition in non–small-cell lung cancer. *New England Journal of Medicine*, 363(18):1693–1703, 2010.

[18] J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009.

[19] Ultan McDermott, Sreenath V Sharma, Lori Dowell, Patricia Greninger, Clara Montagut, Jennifer Lamb, Heidi Archibald, Raul Raudales, Angela Tam, Diana Lee, et al. Identification of genotype-correlated sensitivity to selective kinase inhibitors by using high-throughput tumor cell line profiling. *Proceedings of the National Academy of Sciences*, 104(50):19936–19941, 2007.

[20] Richard M Neve, Koei Chin, Jane Fridlyand, Jennifer Yeh, Frederick L Baehner, Tea Fevr, Laura Clark, Nora Bayani, Jean-Philippe Coppe, Frances Tong, et al. A collection of breast cancer cell lines for the study of functionally distinct cancer subtypes. *Cancer cell*, 10(6):515–527, 2006.

[21] Guillaume Obozinski, Ben Taskar, and Michael Jordan. Joint covariate selection for grouped classification. *Department of Statistics, U. of California, Berkeley, TR*, 743, 2007.

[22] Guillaume Obozinski, Ben Taskar, and Michael I Jordan. Multi-task feature selection. *Statistics Department, UC Berkeley, Tech. Rep*, 2006.

[23] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[24] Charles Sawyers. Targeted cancer therapy. *Nature*, 432(7015):294–297, 2004.

[25] Noah Simon, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. *SGL: Fit a GLM (or cox model) with a combination of lasso and group lasso regularization*, 2013. R package version 1.1.

[26] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

[27] David B Solit, Levi A Garraway, Christine A Pratilas, Ayana Sawai, Gad Getz, Andrea Basso, Qing Ye, Jose M Lobo, Yuhong She, Iman Osman, et al. Braf mutation predicts sensitivity to mek inhibition. *Nature*, 439(7074):358–362, 2005.

[28] Martin L Sos, Kathrin Michel, Thomas Zander, Jonathan Weiss, Peter Frommolt, Martin Peifer, Danan Li, Roland Ullrich, Mirjam Koker, Florian Fischer, et al. Predicting drug susceptibility of non–small cell lung cancers based on genetic lesions. *The Journal of clinical investigation*, 119(6):1727, 2009.

[29] Jane E Staunton, Donna K Slonim, Hilary A Coller, Pablo Tamayo, Michael J Angelo, Johnny Park, Uwe Scherf, Jae K Lee, William O Reinhold, John N Weinstein, et al. Chemosensitivity prediction by transcriptional profiling. *Proceedings of the National Academy of Sciences*, 98(19):10787–10792, 2001.

[30] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[31] John N Weinstein, Timothy G Myers, Patrick M O'Connor, Stephen H Friend, Albert J Fornace, Kurt W Kohn, Tito Fojo, Susan E Bates, Lawrence V Rubinstein, N Leigh Anderson, et al. An information-intensive approach to the molecular pharmacology of cancer. *Science*, 275(5298):343–349, 1997.

[32] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Identification of genomic markers for drug sensitivity using Multi-task Learning

Supplementary information

Nanne Aben[1,2]

[1]Delft Bioinformatics Lab, Delft University of Technology, 2628 CD Delft, The Netherlands
[2]Bioinformatics and Statistics, Department of Molecular Carcinogenesis, 1066 CX Amsterdam, Netherlands Cancer Institute, The Netherlands

1 Supplementary methods

1.1 Discretization of drug sensitivity values

In the data set from Garnett et al. [6], drug response is quantitatively expressed using the IC50, the concentration of the drug at which half of the initial cell population survives. Garnett et al. have fitted a regression model to predict the exact values of these IC50s. However, for cell lines recessive to the drug under investigation, the maximum concentration at which the size of the cell population was measured, was usually lower than their IC50. In those cases, extrapolations had to be used to determine the IC50. These extrapolations are highly inaccurate, so instead of using regression to fit a model to the exact value of this inaccurate result, we have discretized the IC50s into two classes: sensitive and resistant. For the discretization, an outlier detection scheme was used, as proposed by Theo Knijnenburg. This algorithm was unpublished at the moment of writing.

1.2 Dealing with missing values in cluster analysis of drugs

For most drugs, the drug sensitivity has not been tested in all cell lines. In fact, 30% of the drug response values are missing, prohibiting techniques like imputation. Alternatively, excluding all cell lines for which the drug sensitivity to at least one drug is missing is also not an option, since it would leave us with only 107 of the 661 cell lines. Therefore, for the cluster analysis, we have taken the following approach. For each pair of drugs, we considered the cell lines for which drug response was measured in both drugs and computed the correlation in drug response over those cell lines. The resulting $q \times q$ matrix of correlations between drugs was used as a similarity matrix, allowing us to apply hierarchical clustering.

1.3 Dealing with missing values in the Group Lasso constraints

Recall that in Multi-task Group Lasso, we defined an augmented matrix $\mathbf{X}^*$, which replicates the input matrix $\mathbf{X}$ for every drug that is considered in the analysis.

$$\mathbf{X}^* = \begin{bmatrix} \mathbf{X} & 0 & \dots & 0 \\ 0 & \mathbf{X} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mathbf{X} \end{bmatrix} \tag{1}$$

However, for most drugs, the drug sensitivity has not been tested in every cell line. Therefore, for each drug i, we define a matrix $\mathbf{X}_{(i)}$ and a vector $\mathbf{y}_{(i)}$ which contain only the cell lines in which the drug sensitivity has been tested. We now redefine $\mathbf{X}^*$ using these $\mathbf{X}_{(i)}$

$$\mathbf{X}^* = \begin{bmatrix} \mathbf{X}_{(1)} & 0 & \dots & 0 \\ 0 & \mathbf{X}_{(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mathbf{X}_{(q)} \end{bmatrix} \tag{2}$$

And similarly, we redefine $\mathbf{y}^*$ such that it will contain only the cell lines in which the drug sensitivity has been tested.

1.4 Cross-validation

1.4.1 Feature selection

In order to get an unbiased estimate of the performance, the feature selection is always performed within the cross-validation loop. The resulting estimates of performance are then used to select the optimal value for the hyper-parameters, such as the number of features in Forward Feature Selection and λ in the Group Lasso. Next, the model is once more fitted on the entire data set to generate a list of features that would be selected using the optimal hyper-parameter. For the artificial data set experiment, this list is used to measure feature TPs (correctly selected features) and feature FPs (incorrectly selected features). For the drug sensitivity screen analysis, this list is used to create the Venn diagrams.

1.4.2 Division into folds

For the Multi-task Learning methods, we make sure that the division into folds is identical over all tasks. Otherwise a bias would be introduced, as test and train sets would be mixed over the different tasks. One problem occurs if the drug sensitivity value of a certain cell line is not available for a certain drug. We decided to simply remove that cell line from the fold for that task, without replacing it. We observe that the size of the resulting folds remains comparable.

1.4.3 Error bars

For the application of Multi-task Learning to the drug sensitivity screen, cross-validation is repeated 10 times using different random seeds. In each repeat, this results in a different division into folds, resulting in a slightly different estimate of the performance. We use these repeats to create errors bars for the determined AUC.

2 Supplementary images

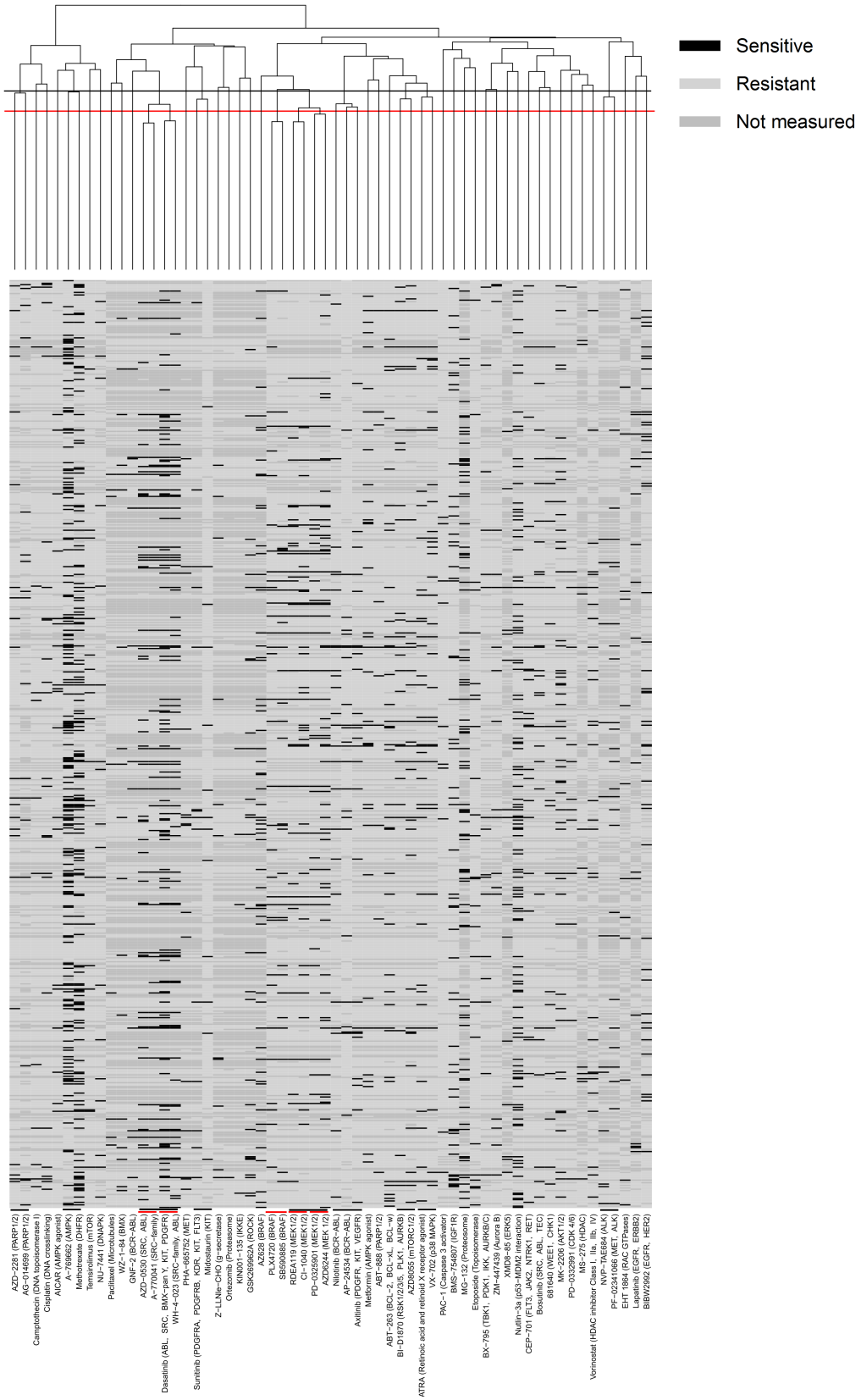


Figure S1: Hierarchical clustering of the drugs in the cell line panel data. The black line and the red line indicate the different cut-offs used for the dendrogram. Resulting clusters are indicated in black and red respectively.

3 Related work during master thesis

3.1 Network-constrained Regularization

Network-constrained Regularization is a tool that can be used to incorporate prior knowledge on pathways in a regression. Even though we did not have sufficient time to apply this method to the drug sensitivity screen, we did include our findings in this thesis, which may be interesting for future work. In Section 3.1.1 we introduce the method as it was originally proposed by Li et al. In Section 3.1.2 we propose an enhanced version of the algorithm, that is able to deal with negative correlations between features. Finally, in Section 3.1.3 we discuss how Network-constrained Regularization could be used for Multi-task Learning, though unfortunately, we found that for Multi-task Learning this method performs rather poor.

3.1.1 Original algorithm

The number of genomic features measured in the drug sensitivity screen is much larger than the number of cell lines in which they are measured, leading to an underdetermined problem. Garnett et al. [6] use the L1- and L2-regularization terms to overcome this problem. However, these regularization terms work solely on the data measured in the cell line panel and cannot utilize prior biological information, such as knowledge on pathways or co-expression. We argue that the use of such prior knowledge can increase predictive performance, reproducibility of the results and interpretability of the selected genotypes.

We propose to incorporate prior knowledge by using Network-constrained Regularization [7]. Given a network that encodes prior knowledge, this method forces regression coefficients $\boldsymbol{\beta}$ to be similar if their corresponding nodes are strongly connected in this network. Network-constrained Regularization differs from Elastic Net regression in that the L2-penalty $\|\boldsymbol{\beta}\|_2$ is replaced by a constraint that models the network-structure:

$$\min_{\boldsymbol{\beta}} \left[\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \boldsymbol{\beta}^T \mathbf{L} \boldsymbol{\beta} \right] \tag{3}$$

Where $\mathbf{L}$ is the graph Laplacian. A definition of the graph Laplacian is omitted here, but let it suffice to state that:

$$\boldsymbol{\beta}^T \mathbf{L} \boldsymbol{\beta} = \sum_{i>j} w_{ij} \left(\frac{\beta_i}{\sqrt{d_i}} - \frac{\beta j}{\sqrt{d_j}} \right)^2 \tag{4}$$

Where w_{ij} indicates the weight on the edge between node i and node j in the biological network; and d_i indicates the degree of node i in the biological network. Inserting this into equation 3 results in:

$$\min_{\boldsymbol{\beta}} \left[\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \sum_{i>j} w_{ij} \left(\frac{\beta_i}{\sqrt{d_i}} - \frac{\beta j}{\sqrt{d_j}} \right)^2 \right] \tag{5}$$

The proposed constraint enforces the regression coefficients $\boldsymbol{\beta}_i$ and $\boldsymbol{\beta}_j$ to be similar when w_{ij} is large, i.e. when the corresponding nodes are strongly connected in the biological network. Li et al. have shown that the resulting optimization problem is convex, as well as that the proposed constraint can replace the L2-penalty for cases where $p \gg n$ and when there are groups of highly correlated variables.

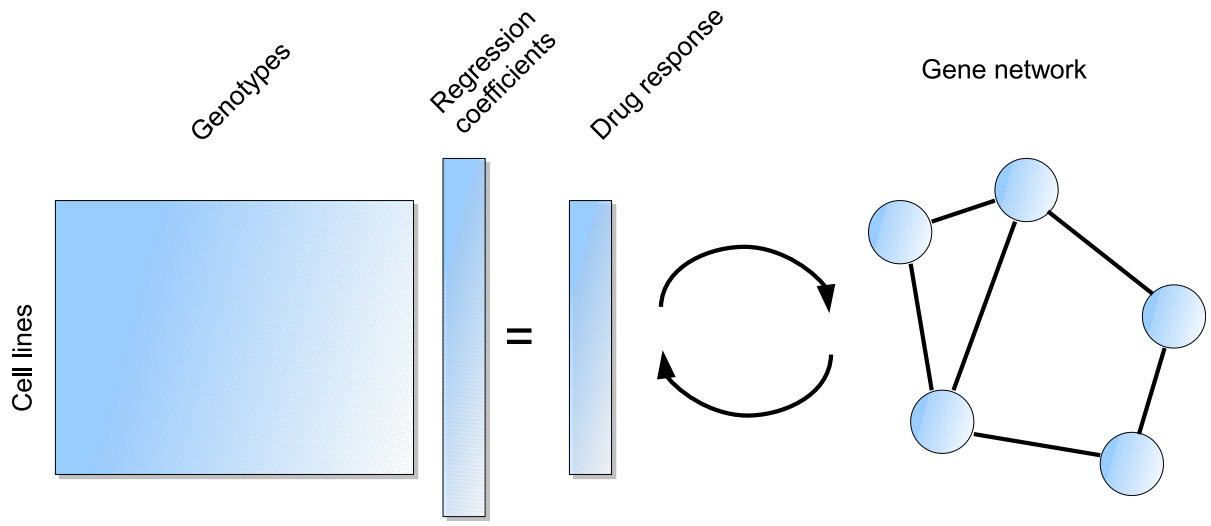


Figure S2: A schematic representation of Network-constrained regularization. On the left, $\mathbf{X}\boldsymbol{\beta} = \mathbf{y}$ is expressed in matrix algebra. On the right, a network encoding prior knowledge is shown. Network-constrained regularization enforces regression coefficients $\boldsymbol{\beta}$ to be similar if their corresponding nodes are strongly connected in this network.

In order to efficiently compute the solution for equation 3, we will rewrite it to a Lasso problem, which can be solved very fast using the LARS algorithm [3]. First we define:

$$\begin{aligned}
\mathbf{L} &= \mathbf{U}\boldsymbol{\Gamma}\mathbf{U}^T \\
\mathbf{S} &= \mathbf{U}\boldsymbol{\Gamma}^{1/2} \\
\mathbf{y}^* &= \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_{p\times 1} \end{bmatrix} \\
\mathbf{X}^* &= \frac{1}{1+\lambda_2}\begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda_2}\mathbf{S}^T \end{bmatrix} \\
\boldsymbol{\beta}^* &= \sqrt{1+\lambda_2} \\
\gamma &= \frac{\lambda_1}{\sqrt{1+\lambda_2}}
\end{aligned} \tag{6}$$

We show that inserting these into a Lasso optimization problem results in a network-regularized opti-

mization.

$$
\begin{aligned}
&\min_{\boldsymbol{\beta}^*}\left[\|\mathbf{y}^* - \mathbf{X}^*\boldsymbol{\beta}^*\|_2^2 + \gamma\|\boldsymbol{\beta}^*\|_1\right] \\
=&\min_{\boldsymbol{\beta}}\left[\left\|\begin{bmatrix}\mathbf{y}\\ \mathbf{0}_{p\times 1}\end{bmatrix} - \frac{1}{1+\lambda_2}\begin{bmatrix}X\\ \sqrt{\lambda_2}S^T\end{bmatrix}\sqrt{1+\lambda_2}\boldsymbol{\beta}\right\|_2^2 + \frac{\lambda_1}{\sqrt{1+\lambda_2}}\left\|\sqrt{1+\lambda_2}\boldsymbol{\beta}\right\|_1\right] \\
=&\min_{\boldsymbol{\beta}}\left[\left\|\begin{bmatrix}\mathbf{y}\\ \mathbf{0}_{p\times 1}\end{bmatrix} - \begin{bmatrix}X\\ \sqrt{\lambda_2}\mathbf{S}^T\end{bmatrix}\boldsymbol{\beta}\right\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1\right] \\
=&\min_{\boldsymbol{\beta}}\left[\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \left\|0_{p\times 1} - \sqrt{\lambda_2}\mathbf{S}^T\boldsymbol{\beta}\right\|_2^2\right] \\
=&\min_{\boldsymbol{\beta}}\left[\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \left\|-\sqrt{\lambda_2}\mathbf{S}^T\boldsymbol{\beta}\right\|_2^2\right] \\
=&\min_{\boldsymbol{\beta}}\left[\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\boldsymbol{\beta}^T\mathbf{S}^T\mathbf{S}\boldsymbol{\beta}\right] \\
=&\min_{\boldsymbol{\beta}}\left[\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\boldsymbol{\beta}^T\mathbf{L}\boldsymbol{\beta}\right] \qquad (7)
\end{aligned}
$$

3.1.2 Enhanced algorithm for application to pathway constraints

One problem with the original algorithm proposed by Li et al. occurs when β_i and β_j are similar to each other, but have an opposite sign. Such a situation is quite likely to occur in biology, for example when gene i is a repressor of gene j. As a solution, we propose to change the constraint to

$$\sum_{i>j} w_{ij}\left(\frac{|\beta_i|}{\sqrt{d_i}} - \frac{|\beta_j|}{\sqrt{d_j}}\right)^2 \qquad (8)$$

It can be shown that this constraint is also convex and that it can be rewritten to a Lasso problem, allowing quick computation. The trick is to split $\boldsymbol{\beta}$ into a positive part $\boldsymbol{\beta}^+$ and a negative part $\boldsymbol{\beta}^-$, such that $\boldsymbol{\beta} = \boldsymbol{\beta}^+ - \boldsymbol{\beta}^-$ and $|\boldsymbol{\beta}| = \boldsymbol{\beta}^+ + \boldsymbol{\beta}^-$.

Consider $\mathbf{X}$ and $\mathbf{y}$ as inputs to the model and define:

$$\mathbf{X}^* = \begin{bmatrix}\mathbf{X} & \text{-}\mathbf{X}\end{bmatrix} \qquad (9)$$

$$\boldsymbol{\beta}^* = \begin{bmatrix}\boldsymbol{\beta}^+ & \boldsymbol{\beta}^-\end{bmatrix} \qquad (10)$$

Add the following constraint:

$$\boldsymbol{\beta}^* \geq 0 \qquad (11)$$

We can map from $\boldsymbol{\beta}^*$ to $|\boldsymbol{\beta}|$ by using

$$\mathbf{C} = \begin{bmatrix}1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0\\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 1\end{bmatrix} \qquad (12)$$

$$|\boldsymbol{\beta}| = \mathbf{C}^T\boldsymbol{\beta}^* \qquad (13)$$

Now define

$$\mathbf{L}^* = \mathbf{C}^T\mathbf{L}\mathbf{C} \qquad (14)$$

Using this, we could rewrite the network-constraint to

$$\boldsymbol{\beta}^{*T}\mathbf{L}^*\boldsymbol{\beta}^* = \sum_{i>j} w_{ij}\left(\frac{(\boldsymbol{\beta}_i^+ + \boldsymbol{\beta}_i^-)}{\sqrt{d_i}} - \frac{(\beta_j^+ + \beta_j^-)}{\sqrt{d_j}}\right)^2 \tag{15}$$

$$= \sum_{i>j} w_{ij}\left(\frac{|\beta_i|}{\sqrt{d_i}} - \frac{|\beta_j|}{\sqrt{d_j}}\right)^2 \tag{16}$$

3.1.3 Proposed algorithm for application to Multi-task Learning

Consider a problem with q different outputs $\mathbf{y}_1 \dots \mathbf{y}_q$. Recall that this can be rewritten from a multi-output regression problem to a regular regression as follows:

$$\|\mathbf{y}^* - \mathbf{X}^*\boldsymbol{\beta}\|_2^2 = \left\| \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_q \end{bmatrix} - \begin{bmatrix} \mathbf{X} & 0 & \cdots & 0 \\ 0 & \mathbf{X} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{X} \end{bmatrix} \boldsymbol{\beta} \right\|_2^2 \tag{17}$$

In order to jointly learn the various outputs, there needs to be some constraint that ties the various outputs together. Consider the following constraint

$$\lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \sum_{f\in\mathcal{F}} \sum_{i>j\in\mathcal{T}} \left(\frac{\beta_{i,f}}{\sqrt{q}} - \frac{\beta_{j,f}}{\sqrt{q}}\right)^2 \tag{18}$$

Where $\mathcal{F}$ is the set of features, $\mathcal{T}$ is the set of tasks and q is the number of tasks. The double-indexing $\beta_{i,f}$ refers to the β corresponding to feature f in task i. This constraint can be obtained using a network-constraint with $\mathbf{L}$ the graph Laplacian of a graph with $pq \times pq$ adjacency matrix:

$$\mathbf{G} = \begin{bmatrix} \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix}_{1,1} & \cdots & \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix}_{q,1} \\ \vdots & \vdots & \vdots \\ \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix}_{1,q} & \cdots & \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix}_{q,q} \end{bmatrix} \tag{19}$$

Where each submatrix is a $p \times p$ identity matrix. Using this adjacency graph, the network constraint ties each feature together over the various outputs. At the same time, the L1-constraint allows the model to select a feature for only a subset of the drugs. Since the problem can be rewritten to a Lasso problem, its solution can usually be computed much faster than when using the Group Lasso, making it a promising candidate for a fast Multi-task Learning algorithm.

Recall from Section 3.1.1 that in order to rewrite Network-constrained Regularization to a Lasso problem, we need to decompose $\mathbf{L}$ into $\mathbf{U}$ and $\boldsymbol{\Gamma}$. For this decomposition, the eigenvalues and the eigenvectors of $\mathbf{L}$ have to be computed. However, since $\mathbf{L}$ is of size $pq \times pq$, it is often too large to calculate naively. Fortunately, there is some structure in $\mathbf{L}$ that can be exploited.

$$\mathbf{L} = \begin{bmatrix} \mathbf{S} & \mathbf{V} & \cdots & \mathbf{V} \\ \mathbf{V} & \mathbf{S} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{V} \\ \mathbf{V} & \cdots & \mathbf{V} & \mathbf{S} \end{bmatrix} \tag{20}$$

Where both $\mathbf{S}$ and $\mathbf{V}$ are $p \times p$ matrices which are defined as:

$$\mathbf{S} = \begin{bmatrix} \frac{q-1}{q} & 0 & \cdots & 0 \\ 0 & \frac{q-1}{q} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{q-1}{q} \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} -\frac{1}{q} & 0 & \cdots & 0 \\ 0 & -\frac{1}{q} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -\frac{1}{q} \end{bmatrix} \tag{21}$$

In order to reduce the amount of space required to represent the full version of L, we will simplify this to

$$\mathbf{L} = \begin{bmatrix} \begin{bmatrix} \frac{q-1}{q} & & \\ & \ddots & \\ & & \frac{q-1}{q} \end{bmatrix}_{1,1} & \cdots & \begin{bmatrix} -\frac{1}{q} & & \\ & \ddots & \\ & & -\frac{1}{q} \end{bmatrix}_{q,1} \\ \vdots & \ddots & \vdots \\ \begin{bmatrix} -\frac{1}{q} & & \\ & \ddots & \\ & & -\frac{1}{q} \end{bmatrix}_{1,q} & \cdots & \begin{bmatrix} \frac{q-1}{q} & & \\ & \ddots & \\ & & \frac{q-1}{q} \end{bmatrix}_{q,q} \end{bmatrix} \tag{22}$$

Recall that we would normally compute the eigenvectors by solving:

$$\mathbf{L}\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{23}$$

Or, in full:

$$\begin{bmatrix} \begin{bmatrix} \frac{q-1}{q} & & \\ & \ddots & \\ & & \frac{q-1}{q} \end{bmatrix}_{1,1} & \cdots & \begin{bmatrix} -\frac{1}{q} & & \\ & \ddots & \\ & & -\frac{1}{q} \end{bmatrix}_{q,1} \\ \vdots & \ddots & \vdots \\ \begin{bmatrix} -\frac{1}{q} & & \\ & \ddots & \\ & & -\frac{1}{q} \end{bmatrix}_{1,q} & \cdots & \begin{bmatrix} \frac{q-1}{q} & & \\ & \ddots & \\ & & \frac{q-1}{q} \end{bmatrix}_{q,q} \end{bmatrix} \begin{bmatrix} v_i^1 \\ \vdots \\ \vdots \\ v_i^p \\ \vdots \\ v_i^{(q-1)p+1} \\ \vdots \\ \vdots \\ v_i^{qp} \end{bmatrix} = \lambda_i \begin{bmatrix} v_i^1 \\ \vdots \\ \vdots \\ v_i^p \\ \vdots \\ v_i^{(q-1)p+1} \\ \vdots \\ \vdots \\ v_i^{qp} \end{bmatrix} \tag{24}$$

Where λ_i is the i-th eigenvalue, v_i is the i-th eigenvector and v_i^j is the j-th element of the i-th eigenvector. Solving the above equation would result in $(q-1)p$ eigenvectors corresponding to $(q-1)p$ eigenvalues. Note that each eigenvector only depends on entries in $\mathbf{L}$ corresponding to a single feature, i.e. eigenvector

i only depends on $\mathbf{L}[\text{ind}, \text{ind}]$ with $\text{ind} = [i : p : ((q-1)p + i)]$. So we could rewrite the decomposition of $\mathbf{L}$ into p decompositions of a $q \times q$ matrix.

$$
\begin{aligned}
&\text{for } i = 1 : q \ \{ \\
&\ \text{ind} = [i : p : ((q-1)p + i)] \\
&\ \mathbf{U}[\text{ind}, \text{ind}] = \text{eigenvectors}(\mathbf{L}[\text{ind}, \text{ind}]) \\
&\ \mathbf{\Gamma}[\text{ind}, \text{ind}] = \text{diag}(\sqrt{\text{eigenvalues}(\mathbf{L}[\text{ind}, \text{ind}])}) \\
&\}
\end{aligned}
$$

The naive algorithm runs in $\mathrm{O}((pq)^3)$, while the above algorithm runs in $\mathrm{O}(pq^3)$, so a speed-up of p^2 is obtained! We can optimize this even further by realizing that each of the p decompositions of $\mathbf{L}[\text{ind}, \text{ind}]$ is in fact the same one. This results in:

$$
\begin{aligned}
&\text{ind} = [1 : p : ((q-1)p + 1)] \\
&\mathbf{L}_{\text{subset}} = L[\text{ind}, \text{ind}] \\
&\text{for } i = 1 : q \ \{ \\
&\ \text{ind} = [i : p : ((q-1)p + i)] \\
&\ \mathbf{U}[\text{ind}, \text{ind}] = \text{eigenvectors}(\mathbf{L}_{\text{subset}}) \\
&\ \mathbf{\Gamma}[\text{ind}, \text{ind}] = \text{diag}(\sqrt{\text{eigenvalues}(\mathbf{L}_{\text{subset}})}) \\
&\}
\end{aligned}
$$

This algorithm has a time-complexity of $\mathrm{O}(q^3)$ and typically finishes within a couple of seconds.

Unfortunately, while this method seemed to have the potential to be a fast Multi-task Learning algorithm, its performance is rather poor. Compared to the Group Lasso, it results in many more spuriously selected features. This can be explained by comparing the two methods. The Group Lasso enforces sparsity between groups. The Sparse Group Lasso extends this, allowing within-group sparsity by adding an external L1-penalty. Network-constrained regularization works slightly different: when a feature is selected, the network-constraint enforces selection of its neighbors in the group (and thus also its neighbors' neighbors, etc.), causing a grouping effect. However, this constraint does not enforce sparsity between groups. Instead, all the sparsity comes from the external L1-penalty. This causes the selection of many small subsets of groups, resulting in increased susceptibility to spurious correlations between features and output.

3.2 Transfer Learning

In a drug sensitivity screen similar to the screen performed by Garnett et al., it was noted by Barretina et al. [1] that because some aberrations only occur commonly in specific tumor types, lineage may have a strong influence on predictive analyses. For example, when applied exclusively to melanoma-derived cell lines, a model trained on melanoma cell lines outperformed a model built on the entire cell line panel.

Even though the case of melanoma illustrates the potential of a lineage-specific model, there are many tissues that are represented by fewer samples as the melanoma cell lines. A model trained on samples from those tissues might perform sub-optimally compared to a model trained on the entire cell line pane, due to the reduced sample size. We argue that lineage-specific models could improve the prediction of genomic markers of drug sensitivity, if the problem of the reduced sample size could somehow be circumvented.

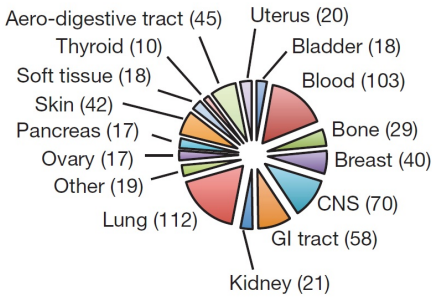


Figure S3: A pie-chart overview of the different lineages present in the cell panel. This Figure was acquired from [6].

We propose to address this issue by using Transfer Learning [8]. This machine learning methodology is based on the idea that it is easier to learn multiple related tasks together rather than learning each task from scratch, provided that there is some common information between the different cell lines. If we define the training of a model for each lineage as a separate task, then Transfer Learning methods can be used to learn these tasks jointly. Joint learning benefits from similarities across tasks, by transferring knowledge between them during training. This allows lineage-specific analysis, while retaining the statistical power of the complete cell line panel, yielding a best-of-both-worlds solution.

There are various approaches on how knowledge should be shared between tasks, for example by transferring knowledge of instances or by transferring knowledge of parameters. In the following sections, we will discuss a method from both of these approaches, as well as their potential application to the identification of genomic markers of drug sensitivity.

3.2.1 Transferring knowledge of instances

When learning a specific task, it is usually not beneficial to use all samples from other tasks, as is illustrated above. However, learning may benefit from certain parts of the data from other tasks. Transferring knowledge of instances is about identifying which parts of the data will benefit the learning of the task at hand.

TraDaBoost [2] is an example of a method that uses this approach. Consider samples from the task at hand (predicting response in a specific tumor cell line type) as the primary data source and consider all other samples as the auxiliary data source. TrAdaBoost starts with all samples from both the primary and auxiliary data sources and then iteratively reweights samples such that the resulting model yields a good performance on the primary data source. For samples the primary data source this method works as regular AdaBoost [5], while for samples from auxiliary data sources it tries to retain only those samples that lead to a good performance on the primary data source. The algorithm can be defined as:

1. Start out with uniform weights on all samples.

2. Draw samples based on their weight.

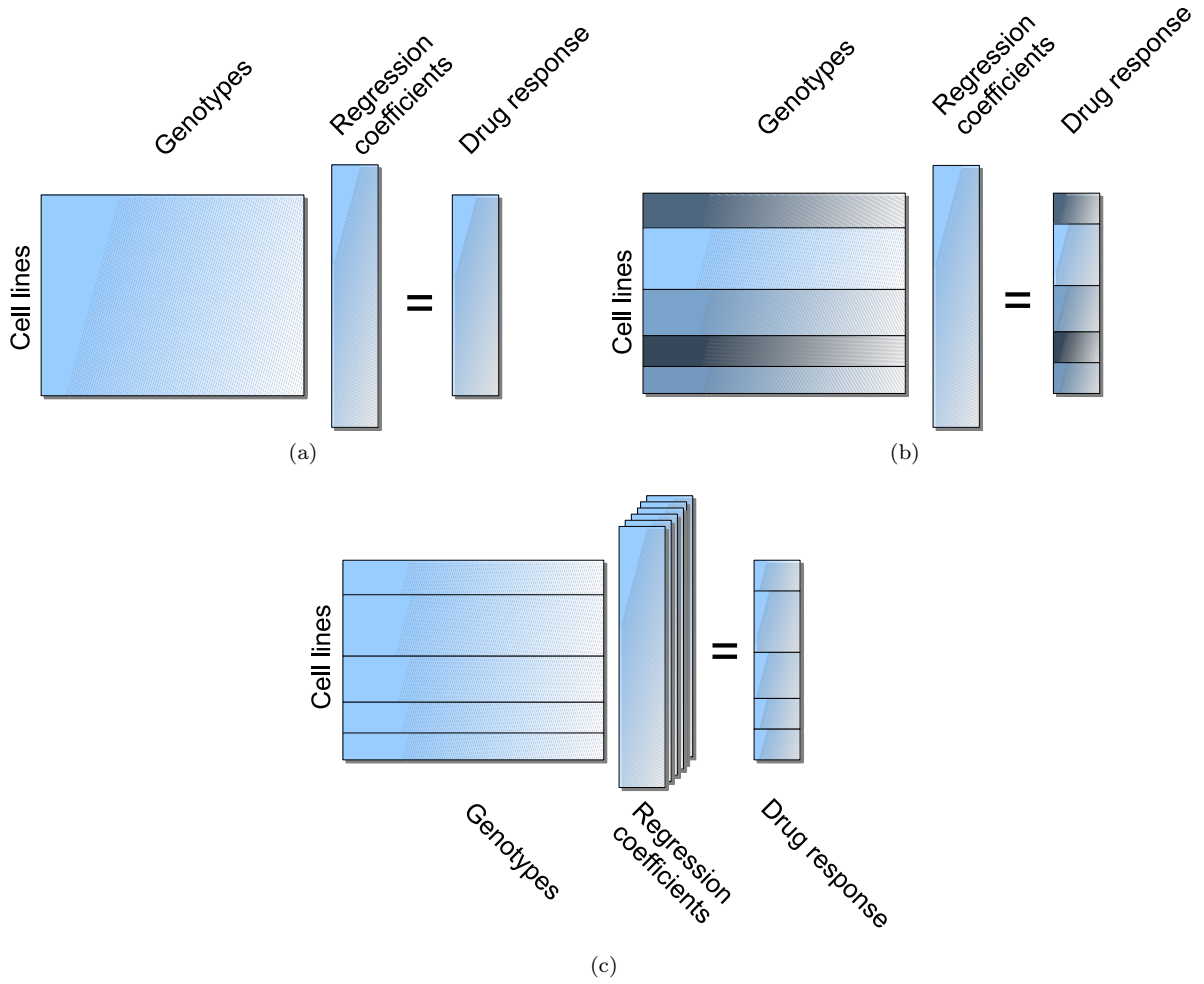


Figure S4: A schematic representation of two Multi-task Learning techniques. **a)** Regular Regression, with $\mathbf{X}\boldsymbol{\beta} = \mathbf{y}$ expressed in matrix algebra. **b)** Transferring knowledge of instances. $\mathbf{X}$ and $\mathbf{y}$ are divided in multiple tasks and a single vector $\boldsymbol{\beta}$ is inferred for the task at hand. Samples from other tasks are used in the regression, but they are weighted based on their relevance to the task at hand. In the image, these weights are indicated by the brightness of the rows. **c)** Transferring knowledge of parameters. $\mathbf{X}$ and $\mathbf{y}$ are again divided in multiple tasks, but in this case six regression coefficient vectors are inferred: a common weight vector $\boldsymbol{\beta}_0$ and task specific weight vectors B_t, $t \in [1 \cdots 5]$.

3. Train a model f_i on the drawn samples.
4. Test the trained model f_i on samples from the primary data source.
5. For each selected sample from the primary data source: if the error is large, increase the sample's weight (as in regular AdaBoost).
6. For each selected sample from the auxiliary data source: if the error is large, decrease the sample's weight (as apparently it is not useful for the task at hand).
7. If number of iterations $i < M$, repeat from Step 2.

The parameter M needs to be optimized using cross-validation. For the model f_i, logistic regression could be used.

3.2.2 Transferring knowledge of parameters

Evgeniou and Pontil [4] have proposed Regularized Multi-task Learning: a method which transfers knowledge of parameters between tasks. Their method is based on the idea that the parameter $\boldsymbol{\beta}$ can be split into two terms for each task: a common term over tasks and a task-specific term. This separation can be written as:

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_0 + \mathbf{b}_t \tag{25}$$

Where $\boldsymbol{\beta}_t$ is the resulting weight for task t, $\boldsymbol{\beta}_0$ is the common weight vector over all tasks and $\mathbf{b}_t$ describes task-specific deviations from $\boldsymbol{\beta}_0$. Though the model is originally proposed for SVMs, it can easily be rewritten to a logistic regression formulation.

$$\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{n} \log\left(1 + e^{-y_i \mathbf{X}_{i*}\boldsymbol{\beta}}\right) + \lambda_1 \left\|\boldsymbol{\beta}_0\right\|_1 + \lambda_2 \left\|\boldsymbol{\beta}_0\right\|_2 + \lambda_3 \left\|\mathbf{b}_t\right\|_1 + \lambda_4 \left\|\mathbf{b}_t\right\|_2 \right] \tag{26}$$

References

[1] Jordi Barretina, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A Margolin, Sungjoon Kim, Christopher J Wilson, Joseph Lehár, Gregory V Kryukov, Dmitriy Sonkin, et al. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603–607, 2012.

[2] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.

[3] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[4] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[5] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[6] Mathew J Garnett, Elena J Edelman, Sonja J Heidorn, Chris D Greenman, Anahita Dastur, King Wai Lau, Patricia Greninger, I Richard Thompson, Xi Luo, Jorge Soares, et al. Systematic identification of genomic markers of drug sensitivity in cancer cells. *Nature*, 483(7391):570–575, 2012.

[7] Caiyan Li and Hongzhe Li. Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics*, 24(9):1175–1182, 2008.

[8] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.