Stacked penalized logistic regression for selecting views in multi-view learning

Loon, W.S. van; Fokkema, M.; Szabo, B.T., Rooij, M.J. de

Stacked Penalized Logistic Regression for Selecting Views in Multi-View Learning

Wouter van Loon[1], Marjolein Fokkema[1], Botond Szabo[2], and Mark de Rooij[1]

[1]Department of Methodology and Statistics, Leiden University
[2]Mathematical Institute, Leiden University

February 6, 2020

Abstract

In biomedical research, many different types of patient data can be collected, such as various types of omics data and medical imaging modalities. Applying multi-view learning to these different sources of information can increase the accuracy of medical classification models compared with single-view procedures. However, collecting biomedical data can be expensive and/or burdening for patients, so that it is important to reduce the amount of required data collection. It is therefore necessary to develop multi-view learning methods which can accurately identify those views that are most important for prediction.

In recent years, several biomedical studies have used an approach known as multi-view stacking (MVS), where a model is trained on each view separately and the resulting predictions are combined through stacking. In these studies, MVS has been shown to increase classification accuracy. However, the MVS framework can also be used for selecting a subset of important views.

To study the view selection potential of MVS, we develop a special case called stacked penalized logistic regression (StaPLR). Compared with existing view-selection methods, StaPLR can make use of faster optimization algorithms and is easily parallelized. We show that nonnegativity constraints on the parameters of the function which combines the views play an important role in preventing unimportant views from entering the model. We investigate the performance of StaPLR through simulations, and consider two real data examples. We compare the performance of StaPLR with an existing view selection method called the group lasso and observe that, in terms of view selection, StaPLR is often more conservative and has a consistently lower false positive rate.

Accepted for publication in *Information Fusion* at https://doi.org/10.1016/j.inffus.2020.03.007.

1 Introduction

Integrating information from different feature sets describing the same set of objects is known as *multi-view learning* [1, 2, 3]. Such different feature sets (*views*) occur naturally in biomedical research as different types of omics data (e.g. genomics, transcriptomics, proteomics, metabolomics) [3], but also as the same profiling data summarized at different levels [4], or as different gene sets or genetic pathways [5]. In neuroimaging, views may present themselves as different MRI modalities, such as functional MRI and diffusion-weighted MRI [6], but also as different feature sets computed from the same structural image [7]. As there is growing interest in integrating multi-omics and imaging data with other sources of information – electronic health records, patient databases, and even social media, wearables, and games – the abundance of multi-view data in biomedical research can only be expected to increase [8, 9].

One common problem in biomedical research is a high-dimensional joint classification and feature selection problem, where – given different classes of objects – the goal is to identify the features most important for accurate classification [3]. When integrating data from multiple views, a typical approach to this problem is *feature concatenation*: simply aggregating the features from all views into one large feature set and fitting a single model to the complete data [3]. This is also known as *early integration*, as the views are combined before any further processing [10, 11]. Commonly used models for feature selection are generalized linear models (GLMs) with an L_1 penalty on the coefficients [*lasso*; 12], or a mixture of L_1 and L_2 penalties [*elastic net*; 13]. Although these methods can obtain sparse solutions by setting some of the coefficients to zero, they do so without regard to the multi-view structure of the data. This structure is important, as data from a single view is often collected together so that the largest potential savings in time and costs are made by selecting or discarding entire views, rather than individual features. Or, for example, when views correspond to genetic pathways, the most associated gene in a pathway may not necessarily be the best candidate for therapeutic intervention [5], and selection of complete pathways may be preferable to selecting individual genes.

The *group lasso* [14] is an extension of the lasso which places a penalty on the sum of L_2 norms of predefined groups of features, leading to a lasso fit at the view level (i.e. view selection), and shrinkage within views. The group lasso has a single tuning parameter which is typically optimized through cross-validation. It is known, however, that the lasso with prediction-optimal penalty parameter selects too many irrelevant features [15, 16]. Likewise, it can be observed in the simulation study of Yuan and Lin [14] that the group lasso tends to select too many groups. Fitting the group lasso can be slow compared with the regular lasso, as parameter updates are performed block-wise rather than coordinate-wise [17, 18]. Furthermore, if sparsity within views is desired, an additional mixing parameter needs to be optimized [19].

The group lasso can be considered a special case of a more general multi-view learning framework known as *multiple kernel learning* [20, 21]. Multiple kernel learning is one of several approaches to multi-view learning. Other popular approaches include *co-training* style algorithms, which are semi-supervised learning algorithms that use the complementary information in different views to iteratively learn several classifiers that can label observations for each other [22, 23], and *co-regularization* style algorithms that put a penalty term on the disagreement between classifiers trained on different views, thus forcing consensus among the view-specific classifiers [2, 23]. Recently, methods which combine the principles of complementarity and consensus into a single method have been proposed, both in classification (i.e. in multi-view support vector machines [24]) and dimension reduction [25].

Another recently popularized multi-view learning framework is that of *multi-view stacking* (MVS) [26, 27]. In order to resolve the limitations of the group lasso, we propose an alternative approach to the view selection problem based on this MVS framework. MVS is a generalization of *stacking* [28] to multi-view data. In stacking, a pool of learning algorithms (the *base-learners* or first level learners) are fitted to the complete data, and their outputs are combined by another algorithm (the *meta-learner*) to obtain a final prediction. The parameterization of the meta-learner is obtained through training on the cross-validated predictions of the base-learners. Since its inception, stacking has been further studied and expanded upon [29, 30, 31]; a more extensive discussion of stacking is provided by Sesmero et al. [32].

In MVS, a base-learner (or pool of base-learners) is trained on each view separately, and a meta-learner is used to combine the predictions of the view-specific models. MVS is thus a *late integration* [10, 11] approach to multi-view learning. Several biomedical studies have applied methods which can be considered a form of MVS, showing improved prediction accuracy compared with single-view models and feature concatenation [7, 26, 33, 34]. Nevertheless, there is no established standard for choosing the base- and meta-learners, and learners which perform well in terms of prediction accuracy often do so at the expense of interpretability. For example, several studies have used random forests as the meta-learner, often with good results in terms of prediction accuracy [27, 33, 34], but the resulting models are difficult to interpret and do not allow for easy selection of the most important views. The ability of a researcher or practitioner to understand how or why a classifier makes decisions (i.e. the topic of *explainable AI* [35, 36]) is important for many real world applications of machine-assisted decision making, particularly in the medical domain [36, 37].

Although applications of MVS have so far focused solely on improving prediction, it also has potential as a group-wise feature selection method. Unfortunately, no unified theoretical underpinning is available regarding the performance of MVS in terms of either prediction accuracy or feature selection. Some progress has been made in terms of the theoretical analysis of the generalization performance of multi-view learning methods [2], for example, the derivation of PAC-Bayes bounds for co-regularization style algorithms in a two-view setting [38]. However, it is not yet clear how these results extend to a setting with more than two views, or whether a similar approach can be used for the theoretical analysis of the multi-view stacking framework. To better understand the MVS approach we introduce *stacked penalized logistic regression* (StaPLR): a special case of MVS where penalized logistic regression is used for both the base-learners and the meta-learner. StaPLR has several advantages over other combinations of base- and meta-learners: logistic regression models are easy to interpret; with appropriately chosen penalties it can be used to perform view selection and/or feature selection within views; and for L_1 and L_2 penalties the regularization path is fast to compute even for a very large number of features [39]. To perform view selection, StaPLR can be applied with, for example, an L_2 penalty at the base level and an L_1 penalty at the meta-level, forming a late integration alternative to the group lasso.

Of primary interest is whether StaPLR selects the correct views, that is, whether it can separate the views containing signal from those containing only noise. Additionally, it is of interest how the classifiers produced by StaPLR perform in terms of predictive accuracy. The derived results can be used as an indicator of the view selection potential of the general MVS approach.

The rest of this article is structured as follows. In Section 2.1 we discuss the multi-view stacking algorithm. In Section 2.2 we verify the importance of nonnegativity constraints on the parameters of the meta-learner for preventing degenerate behavior in MVS with a broad class of base-learners, including penalized GLMs. In Section 3 we introduce StaPLR as a special case of MVS. In Section 4 we compare, on simulated data, the view selection and classification performance of StaPLR with that of the group lasso, and in Section 5 we apply both methods to two gene expression data sets. In Section 6 we relate our results on the performance of StaPLR to the general MVS framework, and in Section 7 we present our conclusions. Theoretical proofs are given in the Appendix.

2 Multi-View Stacking (MVS)

2.1 The MVS Algorithm

Let us denote by $\boldsymbol{X}^{(1)}, ..., \boldsymbol{X}^{(V)}$ a multi-view data set, with $\boldsymbol{X}^{(v)}$ the $n \times m_v$ matrix of features in view v. Let us denote by $\boldsymbol{y} = (y_1, ..., y_n)^T$ the vector of corresponding outcomes. We define a (supervised) learning algorithm or *learner* A as a function that takes as input a labeled data set and produces as output a learned function $\hat{f}$ mapping input vectors to outcomes.

The MVS procedure was already defined and briefly discussed by Li et al. [26] and Garcia-Ceja et al. [27]. Here, we give a somewhat broader definition of the MVS procedure in Algorithm 1, allowing for general base- and meta-learners, and for multiple learners per view. We denote by $A_{v,b}$ the bth base-learner for view v, with B_v the total number of base-learners for that view. We denote the meta-learner by A_{meta}. Although we consider only a single meta-learner, the procedure could easily be extended by using multiple meta-learners and combining their predictions at even higher levels if desired.

The two key components of training any stacked model are (1) training the base-learners, and (2) training the meta-learner. These can be performed in any order, but in Algorithm 1 we first show the training of the base-learners: for each view $\boldsymbol{X}^{(v)}$, $v = 1, ..., V$, we apply the base-learners $A_{v,1}, ..., A_{v,B_v}$ to all n observations of that view to obtain a set of learned functions $\hat{f}_{v,1}, ..., \hat{f}_{v,B_v}$.

To train the meta-learner, we need to obtain a set of cross-validated predictions for each learned function $\hat{f}_{v,b}$. Therefore, we partition the data into K groups, and denote by $S_1, S_2, ..., S_K$ the K-partition of the index set $\{1, 2, ..., n\}$. For each fold $k = 1, ..., K$, we apply the learner $A_{v,b}$ to the observations which are not in S_k, denoted by $\boldsymbol{X}^{(v)}_{i \notin S_k}$, $\boldsymbol{y}_{i \notin S_k}$. We then apply the learned function $\hat{f}_{v,b,k}$ to the observations in S_k to obtain the corresponding cross-validated predictions. Thus we obtain an n-vector of cross-validated predictions for each view and corresponding base-learner, denoted by $\boldsymbol{z}^{(v,b)}$. We collect these vectors in an $n \times B$ matrix $\boldsymbol{Z}$, where $B = \sum_v B_v$. These cross-validated predictions are then used as the input features for the meta-learner to obtain $\hat{f}_{\text{meta}}$. The final stacked prediction function is then $\hat{f}_{\text{meta}}\big(\hat{f}_{1,1}(\boldsymbol{X}^{(1)}), ..., \hat{f}_{V,B_V}(\boldsymbol{X}^{(V)})\big)$.

It is clear that if the meta-learner is chosen such that it returns sparse models, MVS can be used for view selection. If we choose a single base-learner for each view, the view selection problem is just a feature selection problem involving V features. Compared with feature concatenation, where one has to solve a group-wise feature selection problem involving $\sum_v m_v$ features, this is an easier task. Furthermore, all computations performed on lines 3 and 9 of Algorithm 1 are independent across views, base-learners, and cross-validation folds, and can thus be parallelized to improve the scalability of MVS.

Algorithm 1: Multi-View Stacking

```
   Data: Views $\boldsymbol{X}^{(1)}, \dots, \boldsymbol{X}^{(V)}$ and outcomes $\boldsymbol{y} = (y_1, \dots, y_n)^T$.
1  for $v = 1$ to $V$ do
2      for $b = 1$ to $B_v$ do
3          $\hat{f}_{v,b} = A_{v,b}(\boldsymbol{X}^{(v)}, \boldsymbol{y})$
4      end
5  end
6  for $v = 1$ to $V$ do
7      for $b = 1$ to $B_v$ do
8          for $k = 1$ to $K$ do
9              $\hat{f}_{v,b,k} = A_{v,b}(\boldsymbol{X}^{(v)}_{i \notin S_k}, \boldsymbol{y}_{i \notin S_k})$
10             $\boldsymbol{z}^{(v,b)}_{i \in S_k} = \hat{f}_{v,b,k}(\boldsymbol{X}^{(v)}_{i \in S_k})$
11         end
12     end
13 end
14 $\boldsymbol{Z} = (\boldsymbol{z}^{(1,1)}, \boldsymbol{z}^{(1,2)}, \dots, \boldsymbol{z}^{(1,B_1)}, \boldsymbol{z}^{(2,1)}, \dots, \boldsymbol{z}^{(V,B_V)})$
15 $\hat{f}_{\text{meta}} = A_{\text{meta}}(\boldsymbol{Z}, \boldsymbol{y})$
16 $\hat{\boldsymbol{y}} = \hat{f}_{\text{meta}}\big(\hat{f}_{1,1}(\boldsymbol{X}^{(1)}), \dots, \hat{f}_{1,B_1}(\boldsymbol{X}^{(1)}), \hat{f}_{2,1}(\boldsymbol{X}^{(2)}), \dots, \hat{f}_{V,B_V}(\boldsymbol{X}^{(V)})\big)$
```

2.2 Nonnegativity Constraints

In the context of stacked regression, Breiman [29] suggested to constrain the parameters of the meta-learner to be nonnegative and sum to one, in order to create a so-called interpolating predictor, i.e. to ensure that the predictions of the meta-learner stay within the range $[\min_b f_b(\boldsymbol{x}_i), \max_b f_b(\boldsymbol{x}_i)]$ for all observations $\boldsymbol{x}_i$, $i = 1, \ldots, n$. The sum-to-one constraint proved to be generally unnecessary, but the nonnegativity constraints were crucial in finding the most accurate model combinations [29], a finding corroborated by LeBlanc and Tibshirani [40]. However, in a classification context Ting and Witten [41] found that nonnegativity constraints did not substantially affect classification accuracy.

Here we provide an additional argument in favor of nonnegativity constraints from a view-selection perspective. Consider MVS with a base-learner for which one of the possible learned functions returns a constant prediction, such as the intercept-only model. Such base-learners include L_1- and L_2-penalized GLMs. For penalized base-learners, the tuning parameter is often chosen through cross-validation. If we apply a penalized base-learner to some view which contains only noise (i.e. for each feature in this view the true regression coefficient is zero), then it is likely that the model with the lowest cross-validation error is the intercept-only model.

Now let us partition a view into K groups, and again denote by $S_1, S_2, ..., S_K$ the K-partition of the index set $\{1, 2, ..., n\}$. Assuming that for each fold the fitted model is the linear intercept-only model, the K-fold cross-validated predictor $\boldsymbol{z} = (z_1, ..., z_n)^T$ is given by

$$z_i = \frac{1}{n - |S_k|} \sum_{j \notin S_k} y_j \qquad \text{for all } i \in S_k,\ k = 1, ..., K, \tag{1}$$

where $|S_k|$ denotes the cardinality of the set S_k. Given the intercept-only model, the cross-validated predictor is not a function of the features in the corresponding view. Therefore, this view should ideally obtain a weight of zero in the meta-learner. However, the cross-validated predictor is not independent of the outcome: in view of Lemma 1 the correlation between the cross-validated predictor $\boldsymbol{z}$ and the outcome $\boldsymbol{y}$ is always negative, with the strength of the correlation increasing with the number of folds.

Lemma 1 *Let $\boldsymbol{y} = (y_1, ..., y_n)^T$ be the outcome variable, and let $\boldsymbol{z}$ be the cross-validated predictor as defined in (1). Let $\sigma^2(\boldsymbol{y})$ and $\sigma^2(\boldsymbol{z})$ be the empirical variance of the vector $\boldsymbol{y}$ (i.e. $\sigma^2(\boldsymbol{y}) = (n-1)^{-1} \sum_{j=1}^n (y_j - \bar{y})^2$, with $\bar{y} = n^{-1} \sum_{j=1}^n y_j$) and the empirical variance of $\boldsymbol{z}$ (i.e. $\sigma^2(\boldsymbol{z}) = (n-1)^{-1} \sum_{j=1}^n (z_j - \bar{z})^2$, with $\bar{z} = n^{-1} \sum_{j=1}^n z_j$), respectively. Then the Pearson correlation between $\boldsymbol{y}$ and $\boldsymbol{z}$ is equal to*

$$\rho(\boldsymbol{y}, \boldsymbol{z}) = -\frac{\sum_{k=1}^K \left(\sum_{j \in S_k} (y_j - \bar{y}) \right)^2 / (n - |S_k|)}{(n-1)\sigma(\boldsymbol{y})\sigma(\boldsymbol{z})}.$$

The proof can be found in appendix A.

Corollary 1.1 *In the special case when all folds are of the same size, i.e.* $|S_k| = n/K$,

$$\rho(\boldsymbol{y}, \boldsymbol{z}) = -\frac{(K-1)\sigma(\boldsymbol{z})}{\sigma(\boldsymbol{y})}.$$

Corollary 1.2 *In the special case of leave-one-out cross-validation, i.e.* $K = n$,

$$\rho(\boldsymbol{y}, \boldsymbol{z}) = -1.$$

This negative correlation is an artifact of the cross-validation procedure and can produce misleading results in the meta-learner. Consider MVS with two views, $\boldsymbol{X}^{(1)}$ and $\boldsymbol{X}^{(2)}$, where all features are standard normal, and again a single base-learner for which one of the possible fitted models is the linear intercept-only model. Suppose that in truth, the response only depends on the features in $\boldsymbol{X}^{(2)}$, e.g. $\boldsymbol{y} = \boldsymbol{X}^{(2)}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, with $\boldsymbol{\beta}$ a vector of nonzero regression weights, and errors $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)^T$, with $\epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2)$ for all i, and $\sigma_\epsilon^2 > 0$. Then Lemma 2 shows that it can happen that MVS with a linear meta-learner will select the wrong view.

Lemma 2 *Let* $\hat{f}_1$ *be the linear intercept-only model, with leave-one-out cross-validated predictor* $\boldsymbol{z}^{(1)}$, *such that* $\rho(\boldsymbol{y}, \boldsymbol{z}^{(1)}) = -1$. *Let* $\hat{f}_2$ *be a linear model fitted to* $\boldsymbol{X}^{(2)}$, *with cross-validated predictor* $\boldsymbol{z}^{(2)}$, *such that* $0 < \rho(\boldsymbol{y}, \boldsymbol{z}^{(2)}) < 1$. *Then for the linear meta-learner* $\beta_0 + \beta_1\hat{f}_1(\boldsymbol{X}^{(1)}) + \beta_2\hat{f}_2(\boldsymbol{X}^{(2)})$, *the least-squares parameter estimates are*

$$\hat{\beta}_1 = 1 - n,$$
$$\hat{\beta}_2 = 0.$$

The proof can be found in appendix B.

In Lemma 2 a negative weight is given to $\hat{f}_1$ (the intercept-only model), while $\hat{f}_2$ (the model containing signal) is excluded from the meta-learner. The selected view is $\boldsymbol{X}^{(1)}$, which contains only noise. Estimating the coefficients using L_1- or L_2-penalized estimation with tuning parameter selected through cross-validation does not help since the estimated prediction function described in Lemma 2 has zero cross-validation error. Cross-validation will therefore always select the least-penalized model under consideration, thus providing no meaningful shrinkage of β_1. However, nonnegativity constraints can prevent such degenerate behavior by forcing β_1 to be zero, allowing a nonzero estimate of β_2.

Leave-one-out cross-validation is an extreme case, as for smaller values of K such negative correlations will be lower in magnitude. Furthermore, it should be noted that the cross-validated predictors as described in (1) occur only in the training phase of the multi-view stacking procedure. As can be observed in Algorithm 1, the matrix of cross-validated predictions $\boldsymbol{Z}$ is used to train the meta-learner (line 15), but the final stacked prediction function (line 16) uses the view-specific functions learned from the complete views (lines 1:5). Thus, the final stacked prediction function will not produce the kind of piece-wise constant predictions seen in (1). Nevertheless, the introduced correlations can cause the meta-learner to include superfluous views in the model. From a view-selection perspective this is clearly undesirable.

3 Stacked Penalized Logistic Regression (StaPLR)

3.1 Penalized Logistic Regression

For a binary outcome $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^T \in \{0,1\}^n$, and a feature set $\boldsymbol{X} = (x_{ij}) \in \mathbb{R}^{n \times m}$, the logistic regression model is given by

$$\Pr(y_i = 1|\boldsymbol{x}_i) = \frac{1}{1 + \exp(-\beta_0 - \boldsymbol{\beta}^T \boldsymbol{x}_i)}, \tag{2}$$

where $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})^T$ is the feature vector corresponding to observation i, $\beta_0 \in \mathbb{R}$, and $\boldsymbol{\beta} \in \mathbb{R}^m$. Parameter estimates are typically obtained through maximum likelihood estimation. In penalized estimation, a penalty term on $\boldsymbol{\beta}$ is applied in the optimization problem. We write the intercept β_0 separately, as it is usually not penalized. For example, in the case of an L_2 penalty the parameters are estimated as

$$\hat{\beta}_0, \hat{\boldsymbol{\beta}} = \underset{\beta_0, \boldsymbol{\beta}}{\arg\max} \left\{ \sum_{i=1}^{n} \left[y_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i) - \log(1 + \exp(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i)) \right] - \lambda \|\boldsymbol{\beta}\|_2^2 \right\}, \tag{3}$$

where $\lambda \geq 0$ is a tuning parameter. In the case of an L_1 penalty the rightmost term is replaced by $\lambda \|\boldsymbol{\beta}\|_1$.

A suitable value of λ is generally chosen through cross-validation. First, one defines a set of, say, 100 candidate values of λ. Next, for each value of λ, cross-validation is applied to obtain an estimate of the associated out-of-sample error, and the value of λ with lowest cross-validation error is selected. Finally, (3) is optimized using the complete data and the selected value of λ to obtain the final model. It should be noted that, since the value of λ is chosen such that it minimizes cross-validation error, the associated error estimate most likely underestimates the true out-of-sample error [42]. Using cross-validation to choose the tuning parameter should thus be considered part of the learning process. In order to validate such a model using cross-validation, a double cross-validation is required, where an inner loop used to select the tuning parameter is nested in an outer validation loop [42].

3.2 Stacked Penalized Logistic Regression

We previously defined a (supervised) learner A as a function that takes as input a labeled data set and produces as output a learned function $\hat{f}$ mapping input vectors to outcomes. Thus the procedure of fitting a penalized logistic regression model (including cross-validation for λ) described in section 3.1 can be considered a learner. Such a learner is described in pseudocode in Algorithm 2. We define StaPLR as a special case of MVS where all learners are penalized logistic regression learners. StaPLR thus denotes a special case of Algorithm 1, where all $A_{v,b}$ (lines 3 and 9) and A_{meta} (line 15) are functions of the form described in Algorithm 2. Note that in Algorithm 2, $\boldsymbol{X}$ refers to the first input argument of the learner. When applied as a base-learner in a stacked model, this would be a view $\boldsymbol{X}^{(v)}$, and when applied as the meta-learner, it would be the

Algorithm 2: Pseudocode for a penalized logistic regression learner

```
   Input: A set of features $\boldsymbol{X}$, and binary outcomes $\boldsymbol{y}$.
   Output: A learned function $\hat{f}$.
1  Define a set of candidate tuning parameter values $\boldsymbol{\Lambda}$.
2  Randomly split the data into $K$ cross-validation folds of roughly equal size.
3  foreach $\lambda \in \boldsymbol{\Lambda}$ do
4      for $k = 1$ to $K$ do
5          Optimize the penalized likelihood (e.g. equation (3)) using only the
             observations outside of fold $k$.
6          Apply the trained model to obtain an error for each observation in fold $k$.
7      end
8      Average the cross-validation error across all observations.
9  end
10 Choose $\lambda^*$ to be the value in $\boldsymbol{\Lambda}$ with lowest cross-validation error.
11 Optimize the penalized likelihood using all observations and penalty parameter
     value $\lambda^*$ to obtain a set of parameters $\hat{\beta}_0^{(\lambda^*)}$, $\hat{\boldsymbol{\beta}}^{(\lambda^*)}$.
12 Return the learned function $\hat{f}(\boldsymbol{X}) = 1/(1 + \exp(-\hat{\beta}_0^{(\lambda^*)} - \boldsymbol{X}\hat{\boldsymbol{\beta}}^{(\lambda^*)}))$.
```

matrix of cross-validated predictions $\boldsymbol{Z}$. In the remainder of this article we use a single base-learner with an L_2 penalty which we apply to every view. For the meta-learner we choose an L_1 penalty. This way we induce sparsity at the view level and shrinkage within each view, thus providing the configuration most similar to the group lasso model. We use probabilities rather than hard classifications as input for the meta-learner. These values contain information about the uncertainty of the predictions and were previously found to work better in stacked generalization than hard class labels [41]. In order to preserve this information, and because the predictions of the base-learners are already on a common scale, we do not standardize the inputs to the meta-learner. Parameters are estimated using coordinate descent [39]. For both the base and meta-learner's internal cross-validation loops we use $K = 10$. The set of candidate tuning parameter values Λ is a sequence of 100 values adaptively chosen by the software [39].

We demonstrate in our simulations that the addition of nonnegativity constraints on the parameters of the meta-learner improves the view selection performance of StaPLR. When differentiating between StaPLR with and without nonnegativity constraints we use the notation StaPLR$^+$ and StaPLR$^-$, respectively. In coordinate descent, nonnegativity constraints are easily implemented by simply setting coefficients to zero if they become negative during the update cycle [17, 39].

4 Simulations

In this section we compare, on simulated data, the performance of StaPLR with that of the group lasso. In Subsection 4.1 we investigate the view selection performance of both methods under a number of experimental conditions, and in Subsection 4.2 we evaluate the obtained classifiers in terms of area under the receiver operating characteristic curve (AUC). In Subsection 4.3 we investigate the view selection performance of both methods for larger sample sizes, and in Subsection 4.4 we explore how the number of features in a view affects the view selection performance if the amount of signal strength is kept constant.

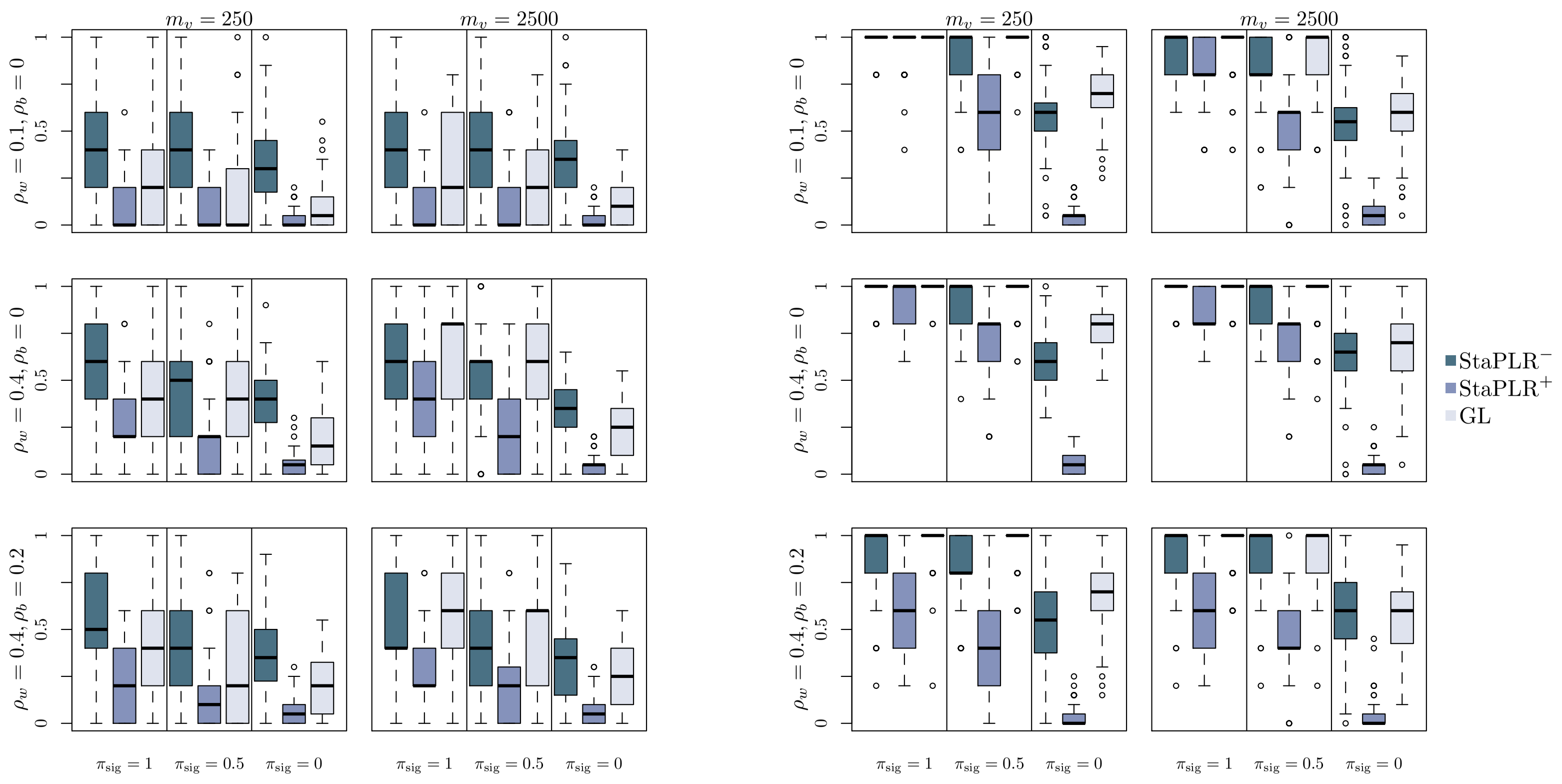
All simulations are performed in R (version 3.4.0) [43]. Penalized logistic regression models are fitted using the package `glmnet` 1.9-8 [39]. The (logistic) group lasso is fitted using the package `gglasso` 1.3 [18].

4.1 View Selection Performance

We investigate the ability of StaPLR and the group lasso to select the correct views. We use two different sample sizes (n = 200 or 2000) and two different view sizes (m_v = 250 or 2500). We use block correlation structures defined by two parameters, namely the population correlation between features in the same view ρ_w, and the population correlation between features in different views ρ_b. We use three different parameterizations: $(\rho_w = 0.1, \rho_b = 0)$, $(\rho_w = 0.4, \rho_b = 0)$, and $(\rho_w = 0.4, \rho_b = 0.2)$, for a total of $2 \times 2 \times 3 = 12$ experimental conditions.

We generate 30 disjoint views of equal size $\boldsymbol{X}^{(v)}$, $v = 1 \ldots 30$, with each view an $n \times m_v$ matrix consisting of normally distributed features scaled to zero mean and unit variance. Within each view, we randomly determine which features correspond to signal (i.e. have a true relation with the response) and which correspond to noise. In 5 views, the probability that a feature corresponds to signal is 1. In another 5 views, the probability that a feature corresponds to signal is 0.5. In the remaining 20 views, the probability that a feature corresponds to signal is 0. Denote by $m = \sum_v m_v$ the total number of features. For each feature $\boldsymbol{x}_j = (x_{1j}, x_{2j}, \ldots, x_{nj})^T$, $j = 1 \ldots m$, we then determine a regression weight θ_j. If $\boldsymbol{x}_j$ corresponds to signal, θ_j = 0.04 or -0.04, each with probability 0.5. This effect size was chosen because simulations showed that with n = 200, m_v = 250, and $(\rho_w = 0.4, \rho_b = 0)$, the class probability distribution is approximately uniform. If $\boldsymbol{x}_j$ corresponds to noise, $\theta_j = 0$. We then determine class probabilities $p_i = 1/(1 + \exp(-\boldsymbol{\theta}^T \boldsymbol{x}_i))$, with $\boldsymbol{\theta} = (\theta_1, \ldots \theta_m)^T$, and class labels $y_i \sim \text{Bernoulli}(p_i)$.

The aim of applying StaPLR and the group lasso is to select the views which contain signal, and discard the others. We calculate the observed probability of a view with a certain proportion of signal (0, 0.5, or 1) being included in the final model. We perform 100 replications per condition. Box plots over all replications are shown in Figure 1. It can be observed that StaPLR with nonnegativity constraints (StaPLR$^+$) maintains a lower false positive rate than the group lasso regardless of sample size, number of features or correlation structure, with the largest differences seen in the n = 2000 case (Figure 1b). It is, however, also more conservative, selecting fewer views containing signal. Without the nonnegativity constraints, StaPLR$^-$ sometimes has a higher false positive rate than the group lasso, particularly when n is small (Figure 1a).



(a) $n = 200$ **(b)** $n = 2000$

Figure 1: Box plots of the observed inclusion probabilities for views with different proportions of signal (denoted by π_{sig}). The within-view correlation is denoted by ρ_w, the between-view correlation by ρ_b, and the number of features per view by m_v.

4.2 Classification Performance

For each replication of each of the 12 conditions from the previous experiment, we generate a test set of size $n = 1000$. The test set is only used for model evaluation; all model fitting including the selection of tuning parameters is performed using (partitions of) the training set. We calculate the AUC on the test set for each of the three methods. It can be observed in Figure 2 that the different methods have a comparable performance when $n = 2000$ and the features from different views are not correlated. When $n = 200$, or when the features from different views are correlated, the group lasso obtains a slightly higher median AUC.

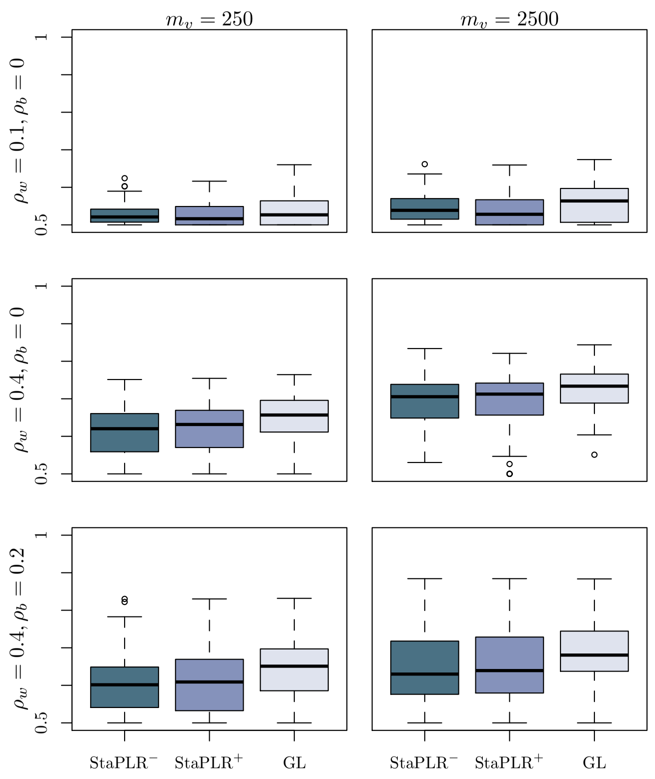
4.3 Larger Sample Sizes

In this experiment we investigate the view selection behavior of StaPLR and the group lasso for larger sample sizes n. We again use 30 views, but now with 25 features per view, and regression weights $\theta_j = 0.12$ or -0.12 if $\boldsymbol{x}_j$ corresponds to signal, to create a smaller problem which is more easily upscaled to larger sample sizes. We consider ten different sample sizes ranging between 50 and 10000, and again calculate the average inclusion probabilities for views with a certain proportion of signal (0, 0.5, or 1). It can be observed in Figure 3 that as n increases, StaPLR$^+$ has an increased probability of selecting views containing signal, while the probability of selecting views containing only noise remains low and even decreases slightly. In contrast, the group lasso and StaPLR$^-$ have an increased probability of selecting both signal and noise views as n increases. Only in some cases for very high values of n is a decrease in the false positive rate observed. StaPLR$^+$ consistently has the lowest false positive rate, although for lower sample sizes it is also generally more conservative, selecting fewer views containing signal. However, at high values of n, StaPLR$^+$ often perfectly distinguishes signal and noise.

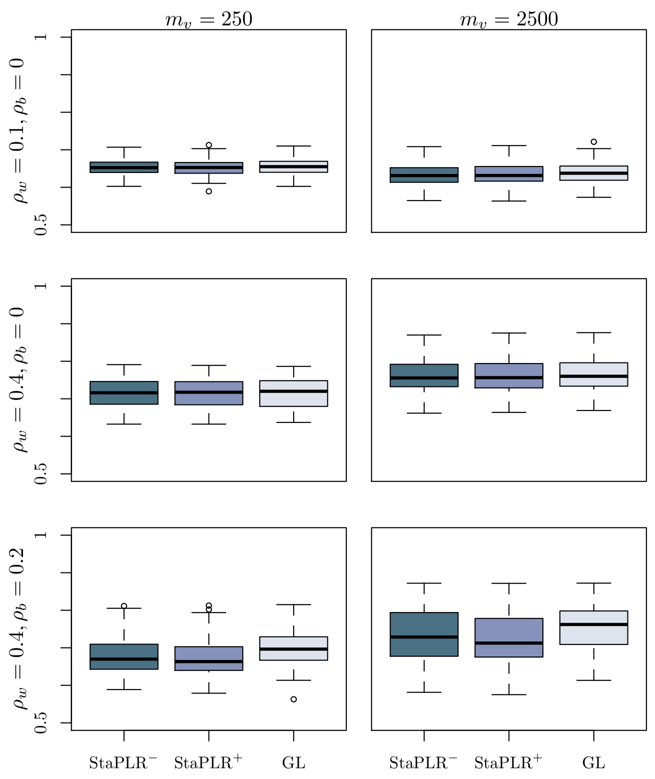
4.4 Different View Sizes

In this experiment we investigate the view selection behavior of StaPLR and the group lasso when views of different sizes are considered at the same time. We consider five different view sizes: 10, 50, 250, 750 and 2500 features. For each view size, we generate one view with signal proportion 1, one view with signal proportion 0.5, and 4 views with signal proportion 0, for a total of 21,360 features across 30 views. We use sample size $n = 2000$. Denote by $\boldsymbol{x}_j^{(v)}$, $j = 1, \ldots, m_v$, the jth feature in view v. If $\boldsymbol{x}_j^{(v)}$ corresponds to signal, its regression weight $\theta_j^{(v)} = 1/\sqrt{m_v}$ or $-1/\sqrt{m_v}$, each with probability 0.5. The rest of the coefficients are set to zero.

The results can be observed in Figure 4. Again, StaPLR$^+$ has the lowest false positive rate, and the inclusion probability of a view containing only noise does not appear to depend on its size. In contrast, the group lasso appears to select large views containing only noise more often than smaller views containing only noise. For views containing signal, it can be observed that larger views are less likely to be included by StaPLR$^+$.



(a) $n = 200$



(b) $n = 2000$

Figure 2: Box plots of the AUC values for 100 test sets per condition. The within-view correlation is denoted by ρ_w, the between-view correlation by ρ_b, and the number of features per view by m_v.

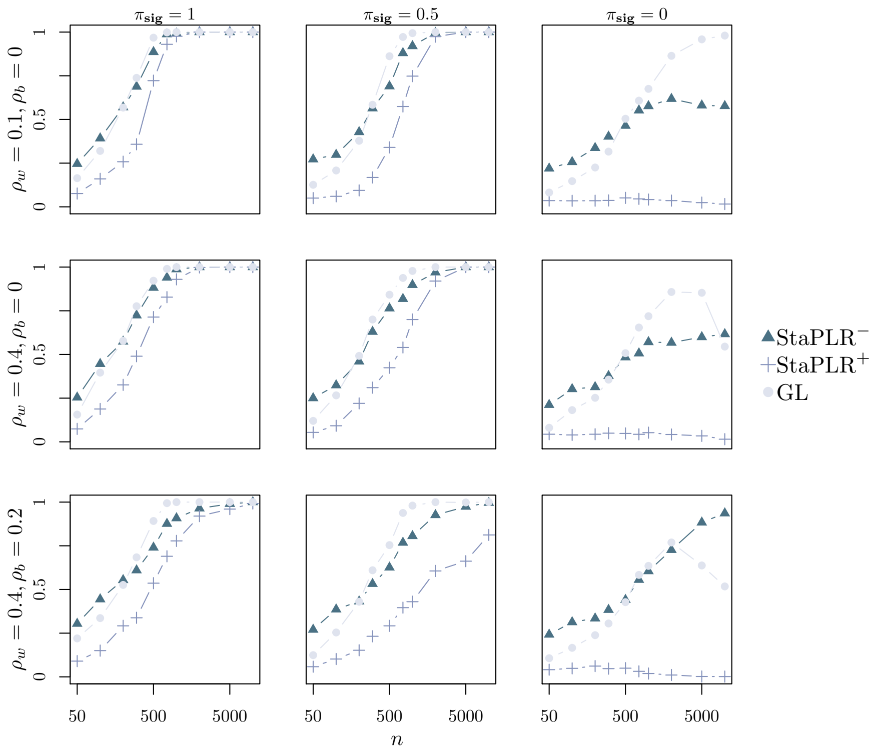


Figure 3: Average inclusion probabilities for views with different proportions of signal (denoted byπ_{sig}), separated by method and correlation structure, across a range of sample sizes. The different sampling points are n = 50, 100, 200, 300, 500, 750, 1000, 2000, 5000 and 10000. Note that the distances along the x-axis are on a $\log_{10}$ scale.

This indicates that for views with the same amount of signal strength in an L_2 sense, StaPLR$^+$ favors views containing less features.

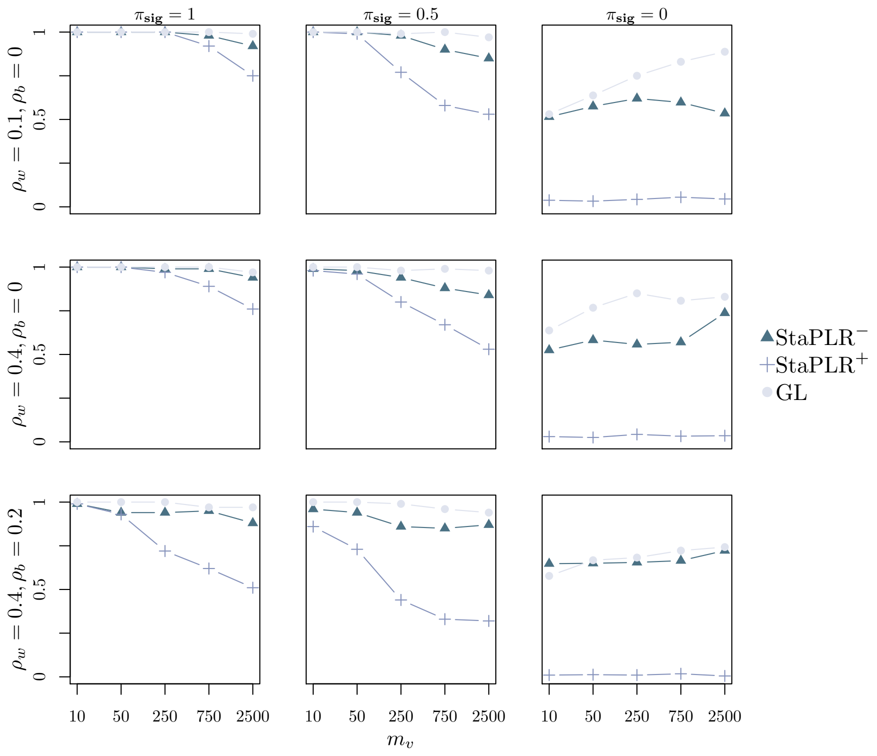


Figure 4: Average inclusion probabilities for views with different proportions of signal (denoted by π_{sig}), separated by method and correlation structure, as a function of view size.

5 Application to Gene Expression Data

One type of multi-view data occurs in gene expression profiling where genes can be divided into gene sets based on, for example, signaling pathway involvement or cytogenetic position [44]. We base our experiments on the real data examples of Simon et al. [19] by applying StaPLR and the group lasso to two gene expression data sets: the colitis data of Burczynski et al. [45], and the breast cancer data of Ma et al. [46].

The colitis data [45] consists of 127 patients: 85 colitis cases (ulcerative colitis or Crohn's disease) and 42 healthy controls. For each patient, gene expression data was collected using an Affymetrix HG-U133A microarray, containing 22,283 probe sets. We matched this data to the C1 cytogenetic gene sets as available from MSigDB 6.1 [44]. We removed any duplicate probes, any genes not included in the C1 gene sets, and any gene sets for which only a single gene was found in the colitis data. Our final feature matrix consisted of 11,761 genes divided across 356 gene sets, with an average of 33 genes per set. All expression levels were $\log_2$-transformed, then standardized to zero mean and unit variance. In Simon et al. [19], the data was randomly split into a training and test set. We apply a similar strategy, randomly splitting the data into two parts of roughly equal size, then using a model fitted to one part to predict the other (i.e. 2-

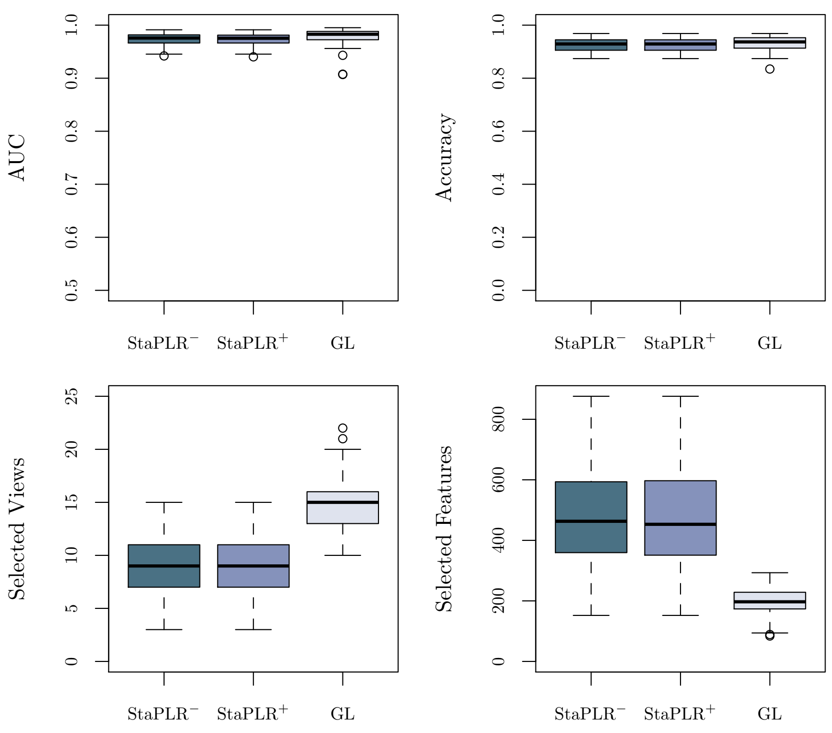


Figure 5: Results of applying StaPLR and group lasso to the colitis gene expression data, in terms of AUC, accuracy, number of selected views and number of selected features.

fold cross-validation). The model fitting includes the selection of all tuning parameters through internal cross-validation loops, and the left-out data is used only for model evaluation. Additionally, we repeat this process 50 times to account for variability due to the random partitioning. We thus obtain 50 sets of predictions for each of the three methods: StaPLR$^-$, StaPLR$^+$, and the group lasso. We calculate both classification accuracy (using a cut-off of .5) and AUC. Additionally, for each of the 50×2 fitted models, we record the number of selected views (gene sets) and features (genes). The results can be observed in Figure 5. All methods have comparable performance in terms of AUC and accuracy, although the group lasso obtains slightly higher median scores than StaPLR. However, StaPLR selects fewer views but more features, whereas the group lasso selects more views but fewer features. The differences between StaPLR$^+$ and StaPLR$^-$ appear negligible.

The breast cancer data [46] consists of 60 tumor samples of patients diagnosed with estrogen positive breast cancer treated with tamoxifen for 5 years, and are labeled according to whether the patients were disease free (32 cases) or cancer recurred (28 cases). For each sample, gene expression data was collected using an Arcturus 22k microarray. We applied the same procedure of matching the gene expression data to the C1 gene sets, obtaining a feature matrix of 12,722 genes divided across 354 sets, with an average

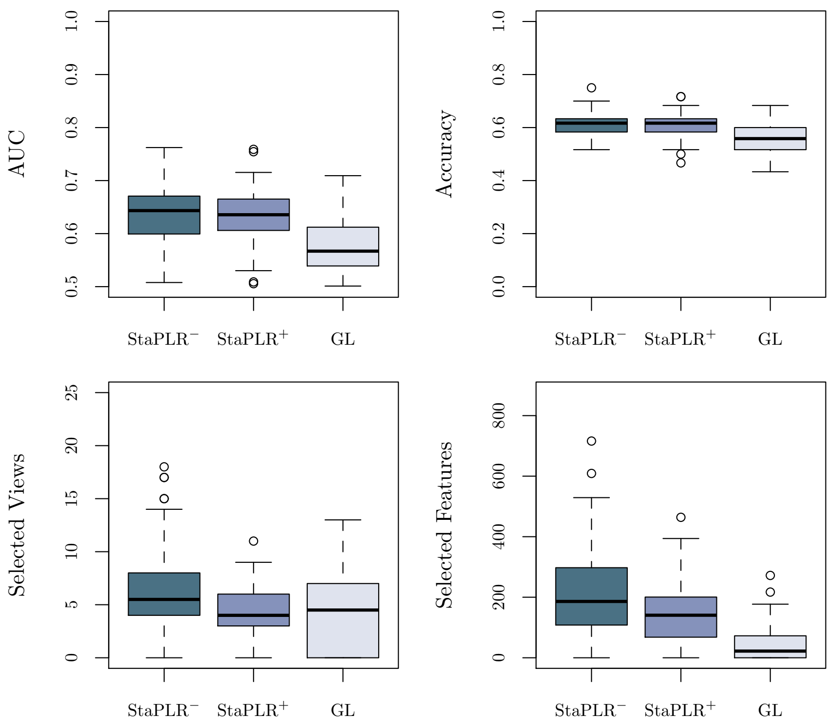


Figure 6: Results of applying StaPLR and group lasso to the breast cancer gene expression data, in terms of AUC, accuracy, number of selected views and number of selected features.

of 36 genes per set. As the data was already $\log_2$-tranformed, we only standardized each feature to zero mean and unit variance. The results can be observed in Figure 6. StaPLR, both with and without nonnegativity constraints, outperforms the group lasso in terms of AUC and accuracy. The differences between StaPLR$^+$ and StaPLR$^-$ in terms of AUC and accuracy are negligible, but the addition of nonnegativity constraints leads to fewer views and fewer features selected on average. Compared with the group lasso, StaPLR$^+$ selects on average a similar number of views, but a larger number of features. However, this is in part caused by the fact that the group lasso selects no views at all in 27% of the produced models, whereas StaPLR$^+$ selects no views in only 4% of the models.

6 Discussion

In our real data examples, StaPLR$^+$ provided similar or better classification accuracy than the group lasso, while selecting a similar or lower number of views. Although the models produced by StaPLR$^+$ in the colitis data were sparser at the view level than those produced by the group lasso, they were not sparser at the feature level: StaPLR$^+$ tended to select a smaller number of views with a larger number of features, while the group lasso tended to select a larger number of views with a smaller number of features. In our simulation experiments on different view sizes (Section 4.4) we observed that StaPLR$^+$ favored views containing fewer features, under the condition that the views contained the same amount of signal strength in an L_2 sense. Although this condition allowed us to investigate the effect of the number of features in isolation from the effect of signal strength, it is unlikely for such a condition to be satisfied in real data. In its presented form, StaPLR does not explicitly favor smaller views, but if such behavior is desired a scaling factor depending on the number of features can easily be added.

In this article we chose a specific parameterization of StaPLR aimed at selecting or discarding entire views. If sparsity within views is desired this could be achieved by, for example, employing an L_1 penalty for the base-learner. This indicates that StaPLR may also form an alternative to other complex penalties such as the sparse group lasso [19].

In our simulations, fitting models using StaPLR was considerably faster than the group lasso although in practice this will depend on the number of views, view size, and available computational resources. As all simulations were performed on a batch scheduling cluster, and such an environment is not ideal for comparing computation speed, we opted not to include a formal speed comparison in our results. However, a plot comparing the computation time of both methods can be observed in Figure 7. Our experience suggests that in the event of a small number of views compared with the number of features per view, a large speed-up can be gained even without any parallelization, by using coordinate-wise rather than block-wise updating. Parallelization can increase this computational speed advantage even further.

The parameter space of StaPLR is very restricted compared with applications of MVS with multiple or more complex base-learners. We chose logistic regression as the base learner since it is among the best-known classifiers in a variety of scientific fields, and because it produces class probabilities rather than simple class labels to use as input for the meta-learner. In a real life setting, one could choose a different base-learner for each view based on domain knowledge, or even try to learn the best base-learner for each view from a set of candidates. More complex base-learners may be able to capture non-linear relationships and increase predictive performance, but this often comes at a cost in terms of model interpretability.

We chose the lasso to perform view selection since it is fast to train and remains a very popular method. Despite its known drawbacks, we found in our experiments that when the lasso was used as the meta-learner in an MVS model with additional nonnegativity constraints, the probability of including superfluous views remained low across all experimental conditions. However, it was generally more conservative than the group lasso, also selecting fewer views containing signal. The performance of the method could potentially be increased by changing the meta-learner. For example, by using a two-step procedure such as the *adaptive lasso* [47], a nonconvex penalty such as SCAD [48], or a subsampling method such as *stability selection* [49]. One could even apply multiple feature-selecting meta-learners and combine their results using some aggregation method [50]. These alternatives can be considerably more computationally

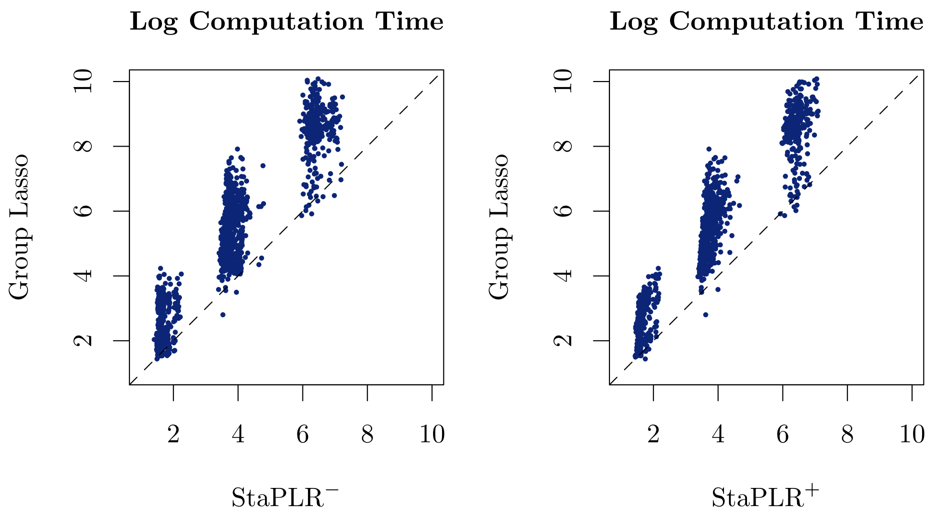


Figure 7: Comparison of the log computation time (in minutes) of StaPLR$^+$ and StaPLR$^-$ with the group lasso. Each point represents one replication of the experiments described in Section 4.1. The different clouds occur due to the different experimental conditions. Note that the base learners of the StaPLR models were fitted sequentially rather than in parallel. In nearly all cases StaPLR$^+$ and StaPLR$^-$ were faster than the group lasso, often considerably so.

expensive than the regular lasso model. However, the use of such methods is still more feasible when applied to the MVS meta-learning problem (which only has a number of features equal to the number of views) than when applied to the full group-wise feature selection problem.

In this article we applied view selection using an MVS-based method in the context of logistic regression. In the future, it would be interesting to investigate how view selection can be applied in the context of other classifiers such as multi-view support vector machines [24], or in the context of multi-view dimension reduction [25].

7 Conclusions

We have introduced stacked penalized logistic regression (StaPLR) as a late integration view selection method based on multi-view stacking. We have further motivated the use of nonnegativity constraints on the parameters of the meta-learner in multi-view stacking with a broad class of base-learners, and shown that such constraints improve the view selection performance of StaPLR. Compared with the group lasso, our simulations have shown that StaPLR with nonnegativity constraints produces sparser models with a comparable, though in some cases slightly reduced, classification performance. This slight reduction in classification performance may be caused by the fact that StaPLR with nonnegativity constraints is often more conservative, also selecting fewer views containing signal than the group lasso. However, it has a much lower false positive rate in terms of the selected views. In our real data examples, StaPLR provided similar or better classification accuracy than the group lasso, while selecting a similar or lower number of views. Our results indicate that multi-view stacking can be used not only to improve model accuracy, but also to select those views that are most important for prediction. By combining view selection with interpretable classifiers we come closer to truly explainable methods for multi-view learning.

A Proof of Lemma 1, Corollary 1.1 and 1.2

The Pearson correlation between the cross-validated predictor $\boldsymbol{z}$ and the outcome $\boldsymbol{y}$ is defined as

$$\rho(\boldsymbol{y},\boldsymbol{z}) = \frac{\sum_{j=1}^{n}(y_j-\bar{y})(z_j-\bar{z})}{(n-1)\sigma(\boldsymbol{y})\sigma(\boldsymbol{z})}.$$

To show that this correlation is always negative we introduce a change of variables. Let $a_j = y_j - \bar{y}$, $j = 1,...,n$, and $\boldsymbol{b} = (b_1, b_2, ..., b_n)$ be the corresponding K-fold cross-validated predictor. Let us denote by b_k^* the value of the predictor in group S_k, $k = 1,...,K$. Note that $b_j = z_j - \bar{y}$ and that $\sigma(\boldsymbol{z}) = \sigma(\boldsymbol{b})$, $\sigma(\boldsymbol{y}) = \sigma(\boldsymbol{a})$. Then

$$\rho(\boldsymbol{y},\boldsymbol{z}) = \rho(\boldsymbol{a},\boldsymbol{b}) = \frac{\sum_{j=1}^{n} a_j(b_j-\bar{b})}{(n-1)\sigma(\boldsymbol{a})\sigma(\boldsymbol{b})} = \frac{\sum_{k=1}^{K}(b_k^*-\bar{b})\sum_{j\in S_k} a_j}{(n-1)\sigma(\boldsymbol{a})\sigma(\boldsymbol{b})}.$$

By noting that $\sum_{k=1}^{K}\sum_{j\in S_k} a_j = \sum_{j=1}^{n} a_j = \sum_{j=1}^{n} y_j - n\bar{y} = 0$ and $\sum_{j\in S_k} a_j = -\sum_{j\notin S_k} a_j$ we get that the numerator on the right hand side of the preceding display is further equal to

$$\begin{aligned}
\sum_{k=1}^{K}(b_k^*-\bar{b})\sum_{j\in S_k} a_j &= \sum_{k=1}^{K}\Big(b_k^*\sum_{j\in S_k} a_j\Big)\\
&= \sum_{k=1}^{K}\Big(\frac{\sum_{j\notin S_k} a_j}{n-|S_k|}\sum_{j\in S_k} a_j\Big)\\
&= -\sum_{k=1}^{K}\frac{(\sum_{j\in S_k} a_j)^2}{n-|S_k|},
\end{aligned}$$

where the term on the right hand side is smaller than or equal to zero and it is exactly zero if in all folds S_k, $k = 1,...,K$, the sum of a_j is zero (i.e. $\sum_{j\in S_k} a_j = 0$). This means that in all folds the average of the observations has to be the same as the total average $\bar{y}$, otherwise the correlation between the vectors will be negative. The final formula for the correlation is then given by

$$\rho(\boldsymbol{y},\boldsymbol{z}) = -\frac{\sum_{k=1}^{K}\Big(\sum_{j\in S_k}(y_j-\bar{y})\Big)^2/(n-|S_k|)}{(n-1)\sigma(\boldsymbol{y})\sigma(\boldsymbol{z})}.$$

Next we consider the special case when all folds are of the same size (i.e. $|S_k| = n/K$, for all $k = 1, ..., K$). Note that in this case $\bar{z} = \bar{y}$, and the formula simplifies to

$$\begin{aligned}
\rho(\boldsymbol{y}, \boldsymbol{z}) &= -\frac{\sum_{k=1}^{K} \Big(n\bar{y} - \sum_{j \notin S_k} y_j - |S_k|\bar{y}\Big)^2 / (n - |S_k|)}{(n-1)\sigma(\boldsymbol{y})\sigma(\boldsymbol{z})} \\
&= -\frac{\sum_{k=1}^{K} (n - |S_k|)\Big(\bar{z} - z_k^*\Big)^2}{(n-1)\sigma(\boldsymbol{y})\sigma(\boldsymbol{z})} \\
&= -\frac{(K-1)\sum_{k=1}^{K} |S_k|\Big(\bar{z} - z_k^*\Big)^2}{(n-1)\sigma(\boldsymbol{y})\sigma(\boldsymbol{z})} \\
&= -\frac{(K-1)\sigma(\boldsymbol{z})}{\sigma(\boldsymbol{y})}.
\end{aligned}$$

In the special case of leave-one-out cross-validation $K = n$. To show that in this case $\rho(\boldsymbol{y}, \boldsymbol{z}) = -1$ it suffices, by the preceding display, to show that $\sigma(\boldsymbol{z}) = \sigma(\boldsymbol{y})/(n-1)$:

$$\begin{aligned}
\sigma^2(\boldsymbol{z}) &= \frac{1}{n-1} \sum_{j=1}^{n} \Big(z_j - \bar{z}\Big)^2 \\
&= \frac{1}{n-1} \sum_{j=1}^{n} \Big(\bar{y}_{(-j)} - \bar{y}\Big)^2 \\
&= \frac{1}{n-1} \sum_{j=1}^{n} \Big(\frac{n\bar{y}}{n-1} - \frac{y_j}{n-1} - \bar{y}\Big)^2 \\
&= \frac{1}{(n-1)^3} \sum_{j=1}^{n} \Big(\bar{y} - y_j\Big)^2 \\
&= \frac{\sigma^2(\boldsymbol{y})}{(n-1)^2}.
\end{aligned}$$

B Proof of Lemma 2

Let $\hat{f}_1$ be the linear intercept-only model, with leave-one-out cross-validated predictor $\boldsymbol{z}^{(1)}$, such that $\rho(\boldsymbol{y}, \boldsymbol{z}^{(1)}) = -1$. Let $\hat{f}_2$ be a second model, with leave-one-out cross-validated predictor $\boldsymbol{z}^{(2)}$, where $0 < \rho(\boldsymbol{y}, \boldsymbol{z}^{(2)}) < 1$. Since $\rho(\boldsymbol{y}, \boldsymbol{z}^{(1)}) = -1$, it follows that $\rho(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}) = -\rho(\boldsymbol{y}, \boldsymbol{z}^{(2)})$.

For the linear meta-learner $\beta_0 + \beta_1 \hat{f}_1(\boldsymbol{X}^{(1)}) + \beta_2 \hat{f}_2(\boldsymbol{X}^{(2)})$, the least-squares parameter estimate of β_1 is given by [51]

$$\hat{\beta}_1 = \frac{1}{\gamma}\Big(\sigma^2(\boldsymbol{z}^{(2)})\text{cov}(\boldsymbol{y}, \boldsymbol{z}^{(1)}) - \text{cov}(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)})\text{cov}(\boldsymbol{y}, \boldsymbol{z}^{(2)})\Big),$$

where $\sigma^2(\cdot)$ denotes the empirical variance, $\text{cov}(\cdot, \cdot)$ denotes the empirical covariance, and $\gamma = (1 - \rho^2(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}))\sigma^2(\boldsymbol{z}^{(1)})\sigma^2(\boldsymbol{z}^{(2)})$. We can rewrite this in terms of correlations as

$$\begin{aligned}
\hat{\beta}_1 &= \frac{1}{\gamma}\Big(\rho(\boldsymbol{y}, \boldsymbol{z}^{(1)}) - \rho(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)})\rho(\boldsymbol{y}, \boldsymbol{z}^{(2)})\Big)\sigma^2(\boldsymbol{z}^{(2)})\sigma(\boldsymbol{z}^{(1)})\sigma(\boldsymbol{y}) \\
&= \frac{(\rho^2(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}) - 1)\sigma^2(\boldsymbol{z}^{(2)})\sigma(\boldsymbol{z}^{(1)})\sigma(\boldsymbol{y})}{(1 - \rho^2(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}))\sigma^2(\boldsymbol{z}^{(2)})\sigma^2(\boldsymbol{z}^{(1)})} \\
&= -\frac{\sigma(\boldsymbol{y})}{\sigma(\boldsymbol{z}^{(1)})} \\
&= -\frac{\sigma(\boldsymbol{y})}{\sigma(\boldsymbol{y})/(n-1)} \\
&= 1 - n.
\end{aligned}$$

Analogously, we can obtain the least-squares estimate of β_2:

$$\begin{aligned}
\hat{\beta}_2 &= \frac{1}{\gamma}\Big(\rho(\boldsymbol{y}, \boldsymbol{z}^{(2)}) - \rho(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)})\rho(\boldsymbol{y}, \boldsymbol{z}^{(1)})\Big)\sigma^2(\boldsymbol{z}^{(1)})\sigma(\boldsymbol{z}^{(2)})\sigma(\boldsymbol{y}) \\
&= \frac{1}{\gamma}\Big(\rho(\boldsymbol{y}, \boldsymbol{z}^{(2)}) - \rho(\boldsymbol{y}, \boldsymbol{z}^{(2)})\Big)\sigma^2(\boldsymbol{z}^{(1)})\sigma(\boldsymbol{z}^{(2)})\sigma(\boldsymbol{y}) \\
&= 0.
\end{aligned}$$

References

[1] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," *arXiv preprint arXiv:1304.5634*, 2013.

[2] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," *Information Fusion*, vol. 38, pp. 43–54, 2017.

[3] Y. Li, F.-X. Wu, and A. Ngom, "A review on machine learning principles for multi-view biological data integration," *Briefings in Bioinformatics*, vol. 19, no. 2, pp. 325–340, 2018.

[4] J. C. Costello, L. M. Heiser, E. Georgii, M. Gönen, M. P. Menden, N. J. Wang, M. Bansal, M. Ammad-Ud-Din, P. Hintsanen, S. A. Khan, J. P. Mpindi, O. Kallioniemi, A. Honkela, T. Aittokallio, K. Wennerberg, J. J. Collins, D. Gallahan, D. Singer, J. Saez-Rodriguez, S. Kaski, J. W. Gray, and G. Stolovitzky, "A community effort to assess and improve drug sensitivity prediction algorithms," *Nature Biotechnology*, vol. 32, no. 12, pp. 1202–1212, 2014.

[5] K. Wang, M. Li, and H. Hakonarson, "Analysing biological pathways in genome-wide association studies," *Nature Reviews Genetics*, vol. 11, no. 12, pp. 843–854, 2010.

[6] M. Fratello, G. Caiazzo, F. Trojsi, A. Russo, G. Tedeschi, R. Tagliaferri, and F. Esposito, "Multi-view ensemble classification of brain connectivity images for neurodegeneration type discrimination," *Neuroinformatics*, vol. 15, no. 2, pp. 199–213, 2017.

[7] F. De Vos, T. Schouten, A. Hafkemeijer, E. Dopper, J. van Swieten, M. de Rooij, J. van der Grond, and S. Rombouts, “Combining multiple anatomical MRI measures improves Alzheimer’s disease classification,” *Human Brain Mapping*, vol. 37, pp. 1920–1929, 2016.

[8] L. Fernández-Luque and T. Bau, “Health and social media: perfect storm of information,” *Healthcare Informatics Research*, vol. 21, no. 2, pp. 67–73, 2015.

[9] C. Auffray, R. Balling, I. Barroso, L. Bencze, M. Benson, J. Bergeron, E. Bernal-Delgado, N. Blomberg, C. Bock, A. Conesa, S. Del Signore, C. Delogne, P. Devilee, A. Di Meglio, M. Eijkemans, P. Flicek, N. Graf, V. Grimm, H. J. Guchelaar, Y. K. Guo, I. G. Gut, A. Hanbury, S. Hanif, R. D. Hilgers, Á. Honrado, D. R. Hose, J. Houwing-Duistermaat, T. Hubbard, S. H. Janacek, H. Karanikas, T. Kievits, M. Kohler, A. Kremer, J. Lanfear, T. Lengauer, E. Maes, T. Meert, W. Müller, D. Nickel, P. Oledzki, B. Pedersen, M. Petkovic, K. Pliakos, M. Rattray, J. R. i Màs, R. Schneider, T. Sengstag, X. Serra-Picamal, W. Spek, L. A. Vaas, O. van Batenburg, M. Vandelaer, P. Varnai, P. Villoslada, J. A. Vizcaíno, J. P. M. Wubbe, and G. Zanetti, “Making sense of big data in health research: towards an EU action plan,” *Genome Medicine*, vol. 8, no. 71, 2016.

[10] W. S. Nobel, “Support vector machine applications in computational biology,” in *Kernel Methods in Computational Biology*, B. Scholkopf, K. Tsuda, and J.-P. Vert, Eds. Cambridge, MA: MIT Press, 2004, ch. 3, pp. 71–92.

[11] M. Zitnik, F. Nguyen, B. Wang, J. Leskovec, A. Goldenberg, and M. M. Hoffman, “Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities,” *Information Fusion*, vol. 50, pp. 71–91, 2019.

[12] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society: Series B*, vol. 58, no. 1, pp. 267–288, 1996.

[13] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *Journal of the Royal Statistical Society: Series B*, vol. 67, no. 2, pp. 301–320, 2005.

[14] M. Yuan and Y. Lin, “Model selection and estimation in regression with grouped variables,” *Journal of the Royal Statistical Society: Series B*, vol. 68, no. 1, pp. 49–67, 2007.

[15] N. Meinshausen and P. Bühlmann, “High-dimensional graphs and variable selection with the lasso.” *Annals of Statistics*, vol. 34, no. 3, pp. 1436–1462, 2006.

[16] A. Benner, M. Zucknick, T. Hielscher, C. Ittrich, and U. Mansmann, “High-dimensional Cox models: the choice of penalty as part of the model building process.” *Biometrical Journal*, vol. 52, no. 1, pp. 50–69, 2010.

[17] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning With Sparsity: The Lasso and Generalizations.* CRC press, 2015.

[18] Y. Yang and H. Zou, "A fast unified algorithm for solving group-lasso penalized learning problems," *Statistics and Computing*, vol. 25, no. 6, pp. 1129–1141, 2015.

[19] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso," *Journal of Computational and Graphical Statistics*, vol. 22, no. 2, pp. 231–245, 2013.

[20] F. R. Bach, G. R. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the smo algorithm," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 6.

[21] F. R. Bach, "Consistency of the group lasso and multiple kernel learning," *Journal of Machine Learning Research*, vol. 9, no. Jun, pp. 1179–1225, 2008.

[22] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational Learning Theory*, 1998, pp. 92–100.

[23] S. Sun, L. Mao, Z. Dong, and L. Wu, *Multiview Machine Learning.* Springer, 2019.

[24] X. Xie and S. Sun, "Multi-view support vector machines with the consensus and complementarity information," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[25] J. Yin and S. Sun, "Multiview uncorrelated locality preserving projection," *IEEE transactions on neural networks and learning systems*, 2019.

[26] R. Li, A. Hapfelmeier, J. Schmidt, R. Perneczky, A. Drzezga, A. Kurz, and S. Kramer, "A case study of stacked multi-view learning in dementia research," in *13th Conference on Artificial Intelligence in Medicine*, 2011, pp. 60–69.

[27] E. Garcia-Ceja, C. E. Galván-Tejada, and R. Brena, "Multi-view stacking for activity recognition with sound and accelerometer data," *Information Fusion*, vol. 40, pp. 45–56, 2018.

[28] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[29] L. Breiman, "Stacked regressions," *Machine Learning*, vol. 24, pp. 49–64, 1996.

[30] M. Van der Laan, E. Polley, and A. Hubbard, "Super learner," *Statistical Applications in Genetics and Molecular Biology*, vol. 6, pp. 49–64, 2007.

[31] S. Sapp, "Subsemble: An ensemble method or combining subset-specific algorithm fits," *Journal of Applied Statistics*, vol. 41, no. 6, pp. 1247–1259, 2014.

[32] M. Sesmero, A. Ledezma, and A. Sanchis, "Generating ensembles of heterogeneous classifiers using Stacked Generalization," *WIREs Data Mining and Knowledge Discovery*, vol. 5, pp. 21–34, 2015.

[33] M. Rahim, B. Thirion, C. Comtat, and G. Varoquaux, "Transmodal learning of functional networks for Alzheimer's disease prediction," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 7, pp. 1204–1213, 2016.

[34] F. Liem, G. Varoquaux, J. Kynast, F. Beyer, S. K. Masouleh, J. M. Huntenburg, L. Lampe, M. Rahim, A. Abraham, R. C. Craddock, S. Riedel-Heller, T. Luck, M. Loeffler, M. L. Schroeter, A. V. Witte, A. Villringer, and D. S. Margulies, "Predicting brain-age from multimodal imaging data captures cognitive impairment," *NeuroImage*, vol. 148, pp. 179–188, 2017.

[35] D. Doran, S. Schulz, and T. Besold, "What does explainable AI really mean? a new conceptualization of perspectives," in *CEUR Workshop Proceedings*, vol. 2071. CEUR, 2018.

[36] A. Holzinger, P. Kieseberg, E. Weippl, and A. M. Tjoa, "Current advances, trends and challenges of machine learning and knowledge extraction: From machine learning to explainable AI," in *Machine Learning and Knowledge Extraction*, ser. Lecture Notes in Computer Science, A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl, Eds., vol. 11015. Cham: Springer, 2018, pp. 1–8.

[37] A. Holzinger, G. Langs, H. Denk, K. Zatloukal, and H. Müller, "Causability and explainability of artificial intelligence in medicine," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 4, p. e1312, 2019.

[38] S. Sun, J. Shawe-Taylor, and L. Mao, "PAC-Bayes analysis of multi-view learning," *Information Fusion*, vol. 35, pp. 117–131, 2017.

[39] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010. [Online]. Available: http://www.jstatsoft.org/v33/i01/

[40] M. LeBlanc and R. Tibshirani, "Combining estimates in regression and classification," *Journal of the American Statistical Association*, vol. 91, no. 436, pp. 1641–1650, 1996.

[41] K. M. Ting and I. H. Witten, "Issues in stacked generalization," *Journal of Artificial Intelligence Research*, vol. 10, pp. 271–289, 1999.

[42] S. Varma and R. Simon, "Bias in error estimation when using cross-validation for model selection," *BMC Bioinformatics*, vol. 7, no. 91, 2006.

[43] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2017. [Online]. Available: https://www.R-project.org/

[44] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P.

Mesirov, “Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles,” *Proceedings of the National Academy of Sciences*, vol. 102, no. 43, pp. 15 545–15 550, 2005.

[45] M. E. Burczynski, R. L. Peterson, N. C. Twine, K. A. Zuberek, B. J. Brodeur, L. Casciotti, V. Maganti, P. S. Reddy, A. Strahs, F. Immermann, W. Spinelli, U. Schwertschlag, A. M. Slager, M. M. Cotreau, and A. J. Dorner, “Molecular classification of Crohn’s disease and ulcerative colitis patients using transcriptional profiles in peripheral blood mononuclear cells,” *The Journal of Molecular Diagnostics*, vol. 8, no. 1, pp. 51–61, 2006.

[46] X.-J. Ma, Z. Wang, P. D. Ryan, S. J. Isakoff, A. Barmettler, A. Fuller, B. Muir, G. Mohapatra, R. Salunga, J. Tuggle, Y. Tran, D. Tran, A. Tassin, P. Amon, W. Wang, W. Wang, E. Enright, K. Stecker, E. Estepa-Sabal, B. Smith, J. Younger, U. Balis, J. Michaelson, A. Bhan, K. Habin, T. M. Baer, J. Brugge, D. A. Haber, M. G. Erlander, and D. C. Sgroi, “A two-gene expression ratio predicts clinical outcome in breast cancer patients treated with tamoxifen,” *Cancer Cell*, vol. 5, no. 6, pp. 607–616, 2004.

[47] H. Zou, “The adaptive lasso and its oracle properties,” *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1418–1429, 2006.

[48] J. Fan and R. Li, “Variable selection via nonconcave penalized likelihood and its oracle properties,” *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001.

[49] N. Meinshausen and P. Bühlmann, “Stability selection,” *Journal of the Royal Statistical Society: Series B*, vol. 72, no. 4, pp. 417–473, 2010.

[50] V. Bolón-Canedo and A. Alonso-Betanzos, “Ensembles for feature selection: A review and future trends,” *Information Fusion*, vol. 52, pp. 1–12, 2019.

[51] J. Fox, *Applied Regression Analysis, Linear Models, and Related Methods*, 2nd ed. Sage Publications, Inc, 2008.