Multi-task Feature Selection Using the Multiple Inclusion Criterion (MIC)

Paramveer S. Dhillon[1], Brian Tomasik[2], Dean Foster[3], and Lyle Ungar[1]

[1] CIS Department, University of Pennsylvania, Philadelphia, PA 19104, U.S.A.
[2] Computer Science Department, Swarthmore College, PA 19081, U.S.A.
[3] Statistics Department, University of Pennsylvania, Philadelphia, PA 19104, U.S.A.

Abstract. We address the problem of joint feature selection in multiple related classification or regression tasks. When doing feature selection with multiple tasks, usually one can "borrow strength" across these tasks to get a more sensitive criterion for deciding which features to select. We propose a novel method, the Multiple Inclusion Criterion (MIC), which modifies stepwise feature selection to more easily select features that are helpful across multiple tasks. Our approach allows each feature to be added to none, some, or all of the tasks. MIC is most beneficial for selecting a small set of predictive features from a large pool of potential features, as is common in genomic and biological datasets. Experimental results on such datasets show that MIC usually outperforms other competing multi-task learning methods not only in terms of accuracy but also by building simpler and more interpretable models.

1 Introduction

We consider the problem of feature selection for a set of related tasks which are expected to partially share common sets of predictive features. When one is trying to predict a set of related responses ("tasks"), be they multiple clinical outcomes for patients or growth rates under different conditions for yeast strains, it may be possible to "borrow strength" by sharing information between the models for the different responses.

The problem of building shared models for multiple related tasks is popularly known as "Multi-Task Learning" or "Transfer Learning" and has been studied extensively [1,2,3,4,5,6,7]. To give a couple examples: [2] do joint empirical risk minimization and treat the multi-response problem by introducing a low-dimensional subspace which is common to all the response variables. [7] construct a multivariate Gaussian prior with a full covariance matrix for a set of "similar" supervised learning tasks and then use semidefinite programming (SDP) to combine these estimates and learn a good prior for the current learning task. Some traditional methods such as neural networks also share parameters between the different tasks [1]. However, none of the above methods does feature selection. This limits their applicability in domains such as genomics, where often only a handful of the thousands of potential features are predictive so that feature selection is very important.

There has been some work on feature selection for multi-task learning. [8] uses maximum-entropy discrimination to select a single subset of features across multiple SVM regression or classification problems that share a common set of potential features. Several other papers work within the framework of regularized regression, taking the penalty term to be an ℓ_1 norm over features of an ℓ_p norm over the coefficients for each feature (an "$\ell_1 - \ell_p$" penalty). [9] consider the case $p = \infty$, while [4,10] use $p = 2$. [4] show that the general subspace selection problem can be formulated as an optimization problem involving the trace norm. [10] focus on the case where the trace norm is not required and instead use a homotopy-based approach to evaluate the entire regularization path efficiently [11]. The idea behind $\ell_1 - \ell_p$ penalties is that when $p > 1$, the cost of making a coefficient nonzero is smaller for features that are shared across more tasks. Indeed, for either $p = 2$ or $p = \infty$, these algorithms tend in practice to yield nonzero coefficients for all of the tasks associated with features that get selected.

Our approach is different. Unlike the above algorithms, we not only select features but select tasks for each feature. This amounts to an $\ell_0 - \ell_0$ penalty ("two-level" sparsity), with one ℓ_0 penalty on features and the other on the associated tasks.[1] Optimization with an exact ℓ_0 penalty requires subset selection, known to be NP-hard [12], but a close solution can be found by stepwise search. Though approximate, stepwise ℓ_0 methods generally yield sparser models than exact ℓ_1 methods [13].

In this paper, we use the information-theoretic Minimum Description Length (MDL) principle [14] to derive an efficient coding scheme for multitask stepwise regression that we call the Multiple Inclusion Criterion (MIC). MIC gives improved performance in prediction for multiple related tasks when there are many candidate features of which only a few are predictive and the predictive features are shared by different tasks. Because it allows each feature to be considered relevant for only a subset of the tasks, MIC achieves sparser models than methods which always share features across all tasks.

The rest of the paper is organized as follows. After outlining the notation used, we describe the MIC coding scheme and its properties in detail. We then present experimental results on synthetic and real datasets, and conclude.

2 Multiple Inclusion Criterion (MIC) for Feature Selection

2.1 Notation Used

The symbols used throughout this section are defined in the Table 1. All the values in the table are given by data except m^*, which is unknown. In particular, we have an $n \times h$ response matrix $\mathbf{Y}$, with a shared $n \times m$ feature matrix $\mathbf{X}$.

[1] Another case of two-level sparsity could be an $\ell_1 - \ell_1$ penalty, but as [10] note, this is equivalent to a single ℓ_1 penalty over the entire set of features and fails to share information across tasks.

Table 1. Symbols used and their definitions

Symbol	Meaning
n	Number of observations
m	Number of candidate features
m^*	Number of beneficial features
h	Total number of tasks
k	Number of tasks into which a feature has been added
j	Index of feature
ν	Index of observation

2.2 A Brief Overview of MIC

In general, penalized likelihood methods like MIC aim to minimize an objective function of the form

$$\text{score} = -\log(\text{likelihood of } \mathbf{Y} \text{ given } \mathbf{X}) + F \times q, \tag{1}$$

where q is the current number of features in the model. As [15] notes, various penalties F have been proposed, including $F = 1$, corresponding to AIC (Akaike Information Criterion), $F = \frac{\ln n}{2}$, corresponding to BIC (Bayesian Information Criterion), and $F = \ln m$, corresponding to RIC (Risk Inflation Criterion—similar to a "Bonferroni correction") [16].

Each of these penalties can be interpreted within the framework of the Minimum Description Length (MDL) principle [14]. MDL envisions a "sender," who knows $\mathbf{X}$ and $\mathbf{Y}$, and a "receiver," who knows only $\mathbf{X}$. In order to transmit $\mathbf{Y}$ using as few bits as possible, the sender encodes not the raw $\mathbf{Y}$ matrix but instead a model for $\mathbf{Y}$ given $\mathbf{X}$, followed by the residuals of $\mathbf{Y}$ about that model. The length S of this message, in bits, is called the *description length* and is the sum of two components. The first is S_E, the number of bits for encoding the residual errors, which according to standard MDL is given by the negative log-likelihood of the data given the model; note that this is the first term of (1). The second component, S_M, is the number of bits used to describe the model itself and can be seen as corresponding to the second term of (1). We use the phrase *total description length* (TDL) to denote the combined length of the message for all h tasks.

In the setting of multiple responses,[2] we select features for the h tasks simultaneously to minimize S. Thus, when we evaluate a feature for addition into the model, we want to maximize the reduction of TDL ΔS^k incurred by adding that feature to a subset k of the h tasks ($1 \leq k \leq h$):

$$\Delta S^k = \Delta S_E^k - \Delta S_M^k$$

[2] We interchangeably use the terms "multiple responses" and "multiple tasks," though our setting is more specific than that of standard multitask problems in which each task may have a different feature matrix; we assume all tasks share the same feature matrix.

where $\Delta S_E^k > 0$ is the reduction in residual-error coding cost due to the data likelihood increase given the new feature, and $\Delta S_M^k > 0$ is the increase in model cost to encode the new feature.[3]

As will be seen in Section 2.3, MIC's model cost includes a component for coding feature coefficients that resembles the AIC or BIC penalty, plus a component for specifying which features are present in the model that resembles the RIC penalty.

In Section 2.4 we compare three approaches to stepwise regression with multiple tasks: (1) a feature is added to all tasks or none; (2) features are added independently to each task (i.e., no transfer); or (3) features can be added to a subset of tasks but the tasks share strength. We first describe a code for approach (3) below.

2.3 Coding Schemes for MIC

Code $\boldsymbol{\Delta S_{jE}^k}$. Let $\mathbf{E}$ be the residual error matrix:

$$\mathbf{E} = \mathbf{Y} - \hat{\mathbf{Y}},$$

where $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ are the $n \times h$ response and prediction matrices, respectively.

ΔS_{jE}^k is the decrease in negative log-likelihood that results from adding feature j to some subset k of the h tasks. If all the tasks were independent, then ΔS_{jE}^k would simply be the sum of the changes in negative log-likelihood for each of the h models separately. However, we may want our model to allow for nonzero covariance among the tasks. This is particularly true for stepwise regression, because in the first iterations of a stepwise algorithm, the effects of features not present in the model show up as part of the "noise" error term, and if two tasks share a feature not yet in the model, the portion of the error term due to that feature will be the same.

Thus, letting ϵ_ν, $\nu = 1, 2, \ldots, n$, denote the error for the νth row of $\mathbf{E}$, we assume $\epsilon_\nu \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \Sigma)$, with Σ an $h \times h$ covariance matrix. In other words,

$$P(\epsilon_\nu) = \frac{1}{\sqrt{(2\pi)^h |\Sigma|}} \exp\left(-\frac{1}{2}\epsilon_\nu^T \Sigma^{-1} \epsilon_\nu\right)$$

in which $(\cdot)^T$, $(\cdot)^{-1}$, and $|\cdot|$ are the matrix transpose, inverse, and determinant, respectively. Therefore,

$$\begin{aligned} S_E &= -\log \prod_{\nu=1}^{n} P(\epsilon_\nu) \\ &= \frac{n}{2}\log\left((2\pi)^h |\Sigma|\right) + \frac{1}{2\ln 2}\sum_{\nu=1}^{n} \epsilon_\nu^T \Sigma^{-1} \epsilon_\nu, \end{aligned} \tag{2}$$

[3] ΔS_E^k is always greater than zero, because even a spurious feature will slightly increase the data likelihood.

where the $\frac{1}{\ln 2}$ factor appears because we use logarithm base 2 (here and throughout the remainder of the paper). Note that the superscript k in ΔS_{jE}^{k} indicates that the reduction is incurred by adding a new feature to k tasks, but the calculation ΔS_{jE}^{k} is over all h tasks; i.e., the whole residual error $\mathbf{E}$ is taken into account.

Code $\boldsymbol{\Delta S_{jM}^{k}}$. To describe ΔS_{jM}^{k} when a feature is added, MIC uses a three-part coding scheme:

$$\Delta S_{jM}^{k} = \ell_I + \ell_H + \ell_{\hat{\theta}},$$

where ℓ_I is the number of bits needed to describe which feature is being added, ℓ_H is the cost of specifying the subset (k) of the h task models in which to include the feature, and $\ell_{\hat{\theta}}$ is the description length of the k nonzero feature coefficients. We now consider different coding schemes for ℓ_I, ℓ_H, and $\ell_{\hat{\theta}}$.

Code ℓ_I. For most data and feature sets, little is known *a priori* about which features will be beneficial.[4] We therefore assume that if a feature x_j is beneficial, its index j is uniformly distributed over $\{1, 2, \ldots, m\}$. This implies $\ell_I = \log m$ bits to encode the index, reminiscent of the RIC penalty for equation (1).

RIC often uses no bits to code the coefficients of the features that are added, based on the assumption that m is so large that the $\log m$ term dominates. This assumption is not valid in the multiple response setting, where the number of models h could be large. If a feature is added to k of the h tasks, the cost of encoding the k coefficients may be a major part of the cost. We describe the cost to code a coefficient below.

Code $\ell_{\hat{\theta}}$. This term corresponds to the number of bits required to code the value of the coefficient of each feature. We could use either AIC or the more conservative BIC to code the coefficients. As explained below, we use 2 bits for each coefficient, similar to AIC.

Given a model, MDL chooses the values of the coefficients that maximize the likelihood of the data. [18] proposes approximating $\hat{\theta}$, the Maximum Likelihood Estimate (MLE), using a grid resolved to the nearest standard error. That is, instead of specifying $\hat{\theta}$, we imagine encoding a rounded-integer value of $\hat{\theta}$'s z-score $\hat{z}$, where $\hat{\theta} = \hat{\theta}_0 + \hat{z}\ \mathrm{SE}(\hat{\theta})$, with $\hat{\theta}_0$ being the default, null-hypothesis value (here, 0) and $\mathrm{SE}(\hat{\theta})$ being the standard error of $\hat{\theta}$.

We assume a "universal prior" distribution for $\hat{z}$, in which half of the probability is devoted to the null value $\hat{\theta}_0$ and the other half is concentrated near $\hat{\theta}_0$ and decays slowly. In particular, for $\hat{\theta} \neq \hat{\theta}_0$, the coding cost is $2 + \log^{+} |\hat{z}|$ $+\ 2 \log^{+} \log^{+} |\hat{z}|$ bits, where $\log^{+}$ denotes logarithm base 2 if the argument is positive, else 0. This prior distribution makes sense in hard problems of feature selection where beneficial features are just marginally significant. Since $\hat{z}$ is quite small in such hard problems, the 2 bits will dominate the other two terms. In fact, we simply assume $\ell_{\hat{\theta}} = 2$.

[4] Following [17], we define a "beneficial" feature as one which, if added to the model, would reduce error on a hypothetical infinite test set.

Code ℓ_H. In order to specify the subset of task models that include a given feature, we encode two pieces of information: First, how many of the tasks (k) have the feature? Second, which subset of k tasks are those?

One way to encode k is to use $\log h$ bits to specify an integer in $\{1, 2, \ldots, h\}$; this implicitly corresponds to a uniform prior distribution on k. However, since we generally expect that smaller values of k are more likely, we instead use coding lengths inspired by the "idealized universal code for the integers" of [19] and [18]: The cost to code k is $\log^* k + c_h$, where $\log^* k = \log^+ k + \log^+ \log^+ k + \log^+ \log^+ \log^+ k + \ldots$, and c_h is the constant required to normalize the implied probability distribution over $\{1, 2, \ldots, h\}$. $c_\infty \approx \log 2.865 \approx 1.516$ [18], but for $h \in \{5, \ldots, 1000\}$, $c_h \approx 1$.

Given k, there are $\binom{h}{k}$ possible subsets of tasks. Taking the subsets to have some pre-specified ordering, we can list the index of our desired subset within that ordering using $\log \binom{h}{k}$ bits.

Thus, in total, we have

$$\ell_H = \log^* k + c_h + \log \binom{h}{k}.$$

2.4 Comparison of the Coding Schemes

The preceding discussion outlined a coding scheme for what we might call "Partially Dependent MIC," or "Partial MIC," in which models for different tasks can share some or all features.

As suggested at the end of Section 2.2, we can also consider a "Fully Dependent MIC," or "Full MIC," scheme in which each feature is shared across all or none of the task models. This amounts to a restricted Partial MIC in which $k = 0$ or $k = h$ for each feature. The advantage comes in not needing to specify the subset of tasks used, saving ℓ_H bits for each feature in the model; however, Full MIC may need to code more coefficient values than Partial MIC.

A third coding scheme is simply to specify each task model in isolation from the others. We call this the "RIC" approach, because each model pays $\log m$ bits for each feature to code its index; this is equivalent, up to the base of the logarithm, to the $F = \ln m$ penalty in equation 1. (However, we include an additional cost of $\ell_{\hat{\theta}}$ bits to code a coefficient.) If the sum of the two costs is sufficiently less than the bits saved by the increase of the data likelihood from adding the feature to the model, the feature will be added to the model. RIC assumes that the beneficial features are not significantly shared across tasks.

We compare the relative coding costs under these three coding schemes for the case where we evaluate a hypothetical feature, x_j, that is beneficial for k tasks and spurious for the remaining $h - k$ tasks. Suppose that Partial MIC and RIC both add the feature to only the k beneficial tasks, while Full MIC adds it to all h tasks. We assume that if the feature is added, the three methods save approximately the same number of bits in encoding residual errors, ΔS_E^k. This would happen if, say, the additional $h - k$ coefficients that Full MIC adds to its models save a negligible number of residual-coding bits (because those features

Table 2. Costs in bits for each of the three schemes to code a model with $k = 1$, $k = \frac{h}{4}$, and $k = h$ nonzero coefficients. $m \gg h \gg 1$, $\ell_I = \log m$, $\ell_{\hat{\theta}} = 2$, and for $h \in \{5, \ldots, 1000\}$, $c_h \approx 1$. Examples of these values for $m = 2{,}000$ and $h = 20$ appear in brackets; the smallest of the costs appears in bold.

k	Partial MIC		Full MIC		RIC	
1	$\log m + c_h + \log h + 2$	[18.4]	$\log m + 2h$	[51.0]	$\log m + 2$	**[13.0]**
$\frac{h}{4}$	$\log m + \log^* \left(\frac{h}{4}\right) + c_h + \log \binom{h}{h/4} + \frac{h}{2}$	**[39.8]**	$\log m + 2h$	[51.0]	$\frac{h}{4} \log m + \frac{h}{2}$	[64.8]
h	$\log m + \log^* h + c_h + 2h$	[59.7]	$\log m + 2h$	**[51.0]**	$h \log m + 2h$	[259.3]

are spurious) and if the estimate for Σ is sufficiently diagonal that the negative log-likelihood calculated using (2) for Partial MIC approximately equals the sum of the negative log-likelihoods that RIC calculates for each response separately.

Table 2 shows that RIC and Partial MIC are the best and the second best coding schemes when $k = 1$, and that their difference is on the order of $\log h$. Full MIC and Partial MIC are the best and the second best coding schemes when $k = h$, and their difference is on the order of $\log^* h$. Partial MIC is best for $k = \frac{h}{4}$.

2.5 Stepwise Search Method

To find a model that minimizes TDL, we use a modified greedy stepwise search as shown in Algorithm 1. For each feature, we evaluate the change in TDL that would result from adding that feature to the model with the optimal number of associated tasks. We add the best feature and then recompute the changes in TDL for the remaining features. This continues until there are no more features that would reduce TDL if added. The number of evaluations of features for possible addition is thus $\mathcal{O}(mm_s)$, where m_s is the number of features eventually added.

To select the optimal number of task models (k) in which to include a given feature, we again use a stepwise-style search. In this case, we evaluate the reduction in TDL that would result from adding the feature to each task, add the feature to the best task, recompute the reduction in TDL for the remaining tasks, and continue.[5] However, unlike a normal stepwise search, we continue this process until we have added the feature to all h task models. The reason for this is two-fold. First, because we want to borrow strength across tasks, we need to avoid overlooking cases where the correlation of a feature with any single task is insufficiently strong to warrant addition, yet the correlations with all of the tasks are. Second, the $\log \binom{h}{k}$ term in Partial MIC's coding cost does not increase monotonically with k, so even if adding the feature to an intermediate number of

[5] A stepwise search that re-evaluates the quality of each task at each iteration is necessary because, if we take the covariance matrix Σ to be nondiagonal, the values of the residuals for one task may affect the likelihood of residuals for other tasks. If we take Σ to be diagonal, as we do in Section 3, then an $\mathcal{O}(h)$ search through the tasks without re-evaluation suffices.

Algorithm 1. Partial MIC

```
1: Include the intercept (feature number 1) in all $h$ response models.
2: $remaining\_features = \{2, \ldots, m\}$.
3: $keep\_adding\_features$ = true.
4: while $keep\_adding\_features$ do
5:    for $j$ in $remaining\_features$ do
6:       // Find the best subset of response models to which to add feature $j$.
7:       for $k = 1$ to $h$ do
8:          Try including feature $j$ in the best $k$ response models. (We greedily assume
            that the best $k$ responses are the union of the best $k-1$ responses with the
            remaining response that, if included, would most increase likelihood.)
9:          Compute $\Delta S^k_{jE}$, the decrease in data residual cost, and $\Delta S^k_{jM}$, the resulting
            increase in model-coding cost, relative to not including feature $j$ in any
            response models.
10:      end for
11:      Let $k_j$ be the value of $k$ that maximizes $\Delta S^k_{jE} - \Delta S^k_{jM}$.
12:      $\Delta S_j := \Delta S^{k_j}_{jE} - \Delta S^{k_j}_{jM}$.
13:   end for
14:   Let $j^*$ be the feature $j$ that maximizes $\Delta S_j$, the reduction in TDL for adding
      feature $j$.
15:   if $\Delta S_{j^*} > 0$ then
16:      Add feature $j^*$ to the appropriate $k_{j^*}$ response models.
17:      $remaining\_features = remaining\_features - \{j^*\}$.
18:   else
19:      $keep\_adding\_features$ = false.
20:   end if
21: end while
```

tasks does not look promising, adding it to all of them might still be worthwhile. Thus, when evaluating a given feature, we compute the description length of the model $\mathcal{O}(h^2)$ times. Since we need to identify the optimal k for each feature evaluation, the entire algorithm requires $\mathcal{O}(h^2 m m_s)$ evaluations of TDL.

While not shown explicitly in Algorithm 1, we use two branch-and-bound-style optimizations to cut this cost significantly in practice:

1. Before searching through subsets of responses to find the optimal subset for each feature, we make an $\mathcal{O}(m)$ sweep through the features to compute an upper bound on the decrease in TDL that could result from adding that feature as

$$\text{(decrease in TDL if the feature is added to all } h \text{ response models)} - \log m. \tag{3}$$

   Here, the first term is an upper bound on the benefit of adding the feature to the optimal number of response models (since adding a feature can only make a model fit better), and the second term underestimates the model cost of adding the feature, regardless of how many response models would actually be used. We sort the features in decreasing order by this upper

bound, and when we reach features whose upper bounds are less than the best actual decrease in TDL observed so far, we terminate the search early.
2. For the stepwise search over responses, we can bound from above the potential benefit of adding the feature to k response models as

$$\begin{aligned}&\text{(decrease in TDL if the feature is added to all } h \text{ response models)}\\&-\left(\log^* k + c_k + \log\binom{h}{k} + 2k\right),\end{aligned} \tag{4}$$

where the subtracted term represents the coding cost of including the feature in k response models. We can stop the search early when no higher value of k has an upper bound that exceeds the best reduction in TDL seen so far for any feature's response subset.[6]

Although we did not attempt to do so, it may be possible to formulate MIC using a *regularization path*, or *homotopy*, algorithm of the sort that have become popular for performing ℓ_1 regularization without the need for cross-validation (e.g., [20]). If possible, this would be significantly faster than stepwise search.

3 Experimental Results

This section evaluates the MIC approach on three synthetic datasets, each of which is designed to match the assumptions of, respectively, the Partial MIC, Full MIC, and RIC coding schemes described in Section 2.4. We also test on two biological data sets, a Yeast Growth dataset [21], which consists of real-valued growth measurements of multiple strains of yeast under different drug conditions, and a Breast Cancer dataset [22], which involves predicting prognosis, ER status, and three other descriptive variables from gene-expression values for different cell lines.

We compare the three coding schemes of Section 2.4 against two other multitask algorithms: "AndoZhang" [2] and "BBLasso" [10], as implemented in the Berkeley Transfer Learning Toolkit [23]. We did not compare MIC with other methods from the toolkit as they all require the data to have additional structure, such as *meta-features* [6,7], or expect the features to be frequency counts, such as for the Hierarchical Dirichlet Processes algorithm. Also, none of the neglected methods does feature selection.

For AndoZhang we use 5-fold CV to find the best value of the parameter that [2] call h (the dimension of the subspace Θ, not to be confused with h as we use it in this paper). We tried values in the range $[1, 100]$ as is done in [2].

MIC as presented in Section 2.3 is a regression algorithm, but AndoZhang and BBLasso are both designed for classification. Therefore, we made each of our responses binary 0/1 values before applying MIC with a regular regression

[6] We say "no higher value of k" rather than "the next higher value of k" because (4) does not decrease monotonically with k, due to the $\log\binom{h}{k}$ quantity.

likelihood term. Once the features were selected, however, we used logistic regression applied to just those features to obtain MIC's actual model coefficients.[7]

As noted in Section 2.3, MIC's negative log-likelihood term can be computed with an arbitrary $h \times h$ covariance matrix Σ among the h tasks. On the data sets in this paper, we found that estimating all h^2 entries of Σ could lead to overfitting, so we instead took Σ to be diagonal. Informal experiments showed that estimating Σ as a convex combination of the full and diagonal estimates could also work well.

3.1 Evaluation on Synthetic Datasets

We created synthetic data according to three separate scenarios—called Partial, Full, and Independent. For each scenario, we generated a matrix of continuous responses as

$$\mathbf{Y}_{n\times h} = \mathbf{X}_{n\times m}\mathbf{w}_{m\times h} + \epsilon_{n\times h}$$

where $m = 2{,}000$ features, $h = 20$ responses, and $n = 100$ observations. Then, to produce binary responses, we set to 1 those response values that were greater than or equal to the average value for their column and set to 0 the rest; this produced a roughly 50-50 split between 1's and 0's because of the normality of the data. Each nonzero entry of w was i.i.d. $\mathcal{N}(0,1)$, and entry of ϵ was i.i.d. $\mathcal{N}(0,0.1)$, with no covariance among the ϵ entries for different tasks. Each task had $m^* = 4$ beneficial features, i.e., each column of w had 4 nonzero entries.

The scenarios differed according to the distribution of the beneficial features in w.

- In the Partial scenario, the first feature was shared across all 20 responses, the second was shared across the first 15 responses, the third across the first 10 responses, and the fourth across the first 5 responses. Because each response had four features, those responses $(6 - 20)$ that did not have all of the first four features had other features randomly distributed among the remaining features (5, 6, ..., 2000).
- In the Full scenario, each response shared exactly features $1 - 4$, with none of features $5 - 2000$ being part of the model.
- In the Independent scenario, each response had four random features among $1, \ldots, 2000$.

For the synthetic data, we report precision and recall to measure the quality of feature selection. This can be done both at a coefficient level (Was each nonzero coefficient in w correctly identified as nonzero, and vice versa?) and at an overall feature level (For features with *any* nonzero coefficients, did we correctly identify them as having nonzero coefficients for any of the tasks, and vice versa?). Note that Full MIC and BBLasso always make entire rows of their estimated w matrices nonzero and so tend to have larger numbers of nonzero coefficients. Table 3 shows the performance of each of the methods on five instances of the

[7] This contrasts with BBLasso, which in the default Transfer Learning Toolkit implementation uses the original regression coefficients (mostly zeros).

Table 3. Test-set accuracy, precision, and recall of MIC and other methods on 5 instances of various synthetic data sets generated as described in Section 3.1. Standard errors are reported over each task; that is, with 5 data sets and 20 tasks per data set, the standard errors represent the sample standard deviation of 100 values divided by $\sqrt{100}$ (except for feature-level results, which apply only to entire data sets and so are divided by $\sqrt{5}$). Baseline accuracy, corresponding to guessing the majority category, is roughly 0.5. *Note:* AndoZhang's NA values are due to the fact that it does not explicitly select features.

Method	True Model	Partial MIC	Full MIC	RIC	BBLasso	AndoZhang
Partial Synthetic Dataset						
Test error	0.07 ± 0.00	$\mathbf{0.10 \pm 0.00}$	0.17 ± 0.01	0.12 ± 0.01	0.19 ± 0.01	0.50 ± 0.02
Coeff. precision	1.00 ± 0.00	0.84 ± 0.02	0.26 ± 0.01	0.84 ± 0.02	0.04 ± 0.00	NA
Coeff. recall	1.00 ± 0.00	0.77 ± 0.02	0.71 ± 0.03	0.56 ± 0.02	0.81 ± 0.02	NA
Feature precision	1.00 ± 0.00	0.99 ± 0.01	0.97 ± 0.02	0.72 ± 0.05	0.20 ± 0.03	NA
Feature recall	1.00 ± 0.00	0.54 ± 0.05	0.32 ± 0.03	0.62 ± 0.04	0.54 ± 0.01	NA
Full Synthetic Dataset						
Test error	0.07 ± 0.00	$\mathbf{0.08 \pm 0.00}$	$\mathbf{0.08 \pm 0.00}$	0.11 ± 0.01	0.09 ± 0.00	0.45 ± 0.02
Coeff. precision	1.00 ± 0.00	0.98 ± 0.01	0.80 ± 0.00	0.86 ± 0.02	0.33 ± 0.03	NA
Coeff. recall	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.63 ± 0.02	1.00 ± 0.00	NA
Feature precision	1.00 ± 0.00	0.80 ± 0.00	0.80 ± 0.00	0.36 ± 0.06	0.33 ± 0.17	NA
Feature recall	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	NA
Independent Synthetic Dataset						
Test error	0.07 ± 0.00	0.17 ± 0.01	0.36 ± 0.01	$\mathbf{0.13 \pm 0.01}$	0.35 ± 0.01	0.49 ± 0.00
Coeff. precision	1.00 ± 0.00	0.95 ± 0.01	0.06 ± 0.01	0.84 ± 0.02	0.02 ± 0.00	NA
Coeff. recall	1.00 ± 0.00	0.44 ± 0.02	0.15 ± 0.02	0.58 ± 0.02	0.43 ± 0.02	NA
Feature precision	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.83 ± 0.02	0.30 ± 0.05	NA
Feature recall	1.00 ± 0.00	0.44 ± 0.02	0.14 ± 0.02	0.58 ± 0.03	0.42 ± 0.06	NA

Partial, Full, and Independent synthetic data sets. On the Partial data set, *Partial MIC* performed the best, closely followed by *RIC*; on the Full synthetic data, *Full MIC* and *Partial MIC* performed equally well; and on the Independent synthetic data, the *RIC* algorithm performed the best closely followed by *Partial MIC*. It is also worth noting that the best-performing methods tended to have the best precision and recall on coefficient selection. The performance trends of the three methods are in consonance with the theory of Section 2.4.

The table shows that only in one of the three cases does one of these methods compete with MIC methods. BBLasso on the Full synthetic data shows comparable performance to the MIC methods, but even in that case it has a very low feature precision, since it added many more spurious features than the MIC methods.

3.2 Evaluation on Real Datasets

This section compares the performance of MIC methods with AndoZhang and BBLasso on a Yeast dataset and Breast Cancer dataset. These are typical of biological datasets in that only a handful of features are predictive from thousands

Table 4. Accuracy and number of coefficients and features selected on five folds of CV for the Yeast and Breast Cancer data sets. For the Yeast data, $h = 20$, $m = 6{,}715$, $n = 104$. For the Breast Cancer data, $h = 5$, $m = 5{,}000$, $n = 100$. Standard errors are over the five CV folds; i.e., they represent (sample standard deviation) / $\sqrt{5}$. *Note:* AndoZhang's NA values are due to the fact that it does not explicitly select features.

Method	Partial MIC	Full MIC	RIC	BBLasso	AndoZhang
			Yeast Dataset		
Test error	**0.38 ± 0.04**	0.39 ± 0.04	0.41 ± 0.05	0.43 ± 0.03	0.39 ± 0.03
Num. coeff. sel.	22 ± 4	64 ± 4	9 ± 1	1268 ± 279	NA
Num. feat. sel.	4 ± 0	3 ± 0	9 ± 1	63 ± 14	NA
			Breast Cancer Dataset		
Test error	**0.33 ± 0.08**	0.37 ± 0.08	0.36 ± 0.08	**0.33 ± 0.08**	0.44 ± 0.03
Num. coeff. sel.	3 ± 0	11 ± 1	2 ± 0	61 ± 19	NA
Num. feat. sel.	2 ± 0	2 ± 0	2 ± 0	12 ± 4	NA

of potential features. This is precisely the case in which MIC outperforms other methods. MIC not only gives better accuracy but does so by choosing fewer features than BBLasso's $\ell_1 - \ell_2$-based approach.

Yeast Dataset. Our Yeast dataset comes from [21]. It consists of real-valued growth measurements of 104 strains of yeast ($n = 104$ observations) under 313 drug conditions. In order to make computations faster, we hierarchically clustered these 313 conditions into 20 groups using correlation as the similarity measure. Taking the average of the values in each cluster produced $h = 20$ real-valued responses (tasks), which we then binarized into two categories: values at least as big as the average for that response (set to 1) and values below the average (set to 0).[8] The features consisted of 526 markers (binary values indicating major or minor allele) and 6,189 transcript levels in rich media for a total of $m = 6{,}715$ features.

Table 4 shows test errors from 5-fold CV on this data set. As can be seen from the table, Partial MIC performs better than BBLasso and AndoZhang. RIC and Full MIC perform slightly worse than Partial MIC, underscoring the point that it is preferable to use a more general MIC coding scheme compared to Full MIC or RIC. The latter methods have strong underlying assumptions, which cannot always correctly capture sharing across tasks. Like Partial MIC, AndoZhang did well on this data set; however, because the algorithm scales poorly with large numbers of tasks, the computation took 39 days (at least using a naïve extension of the default Transfer Learning Toolkit implementation).

Breast Cancer Dataset. Our second data set pertains to Breast Cancer, containing data from five of the seven data sets used in [22]. It contains 1,171

[8] The split was not exactly 50-50, as the data were sometimes skewed, with the mean falling above or below the median.

observations for 22,268 RMA-normalized gene-expression values. We considered five associated responses (tasks); two were binary—prognosis ("good" or "poor") and ER status ("positive" or "negative")—and three were not—age (in years), tumor size (in mm), and grade (1, 2, or 3). We binarized the three non-binary responses into two categories: Response values at least as high as the average, and values below the average. Finally we scaled the dataset down to $n = 100$ and $m = 5{,}000$ (the 5,000 features with the highest variance), to save computational resources. Table 4 shows test errors from 5-fold CV on this data set. As is clear from the table, Partial MIC and BBLasso are the best methods here. But as was the case with other datasets, BBLasso puts in more features, which is undesirable in domains (like biology and medicine) where simpler and hence more interpretable model are sought.

4 Conclusion

We proposed a novel coding scheme, MIC, for feature selection in the presence of multiple related tasks. MIC is a penalized-likelihood method based on the principle of Minimum Description Length (MDL). Sharing across tasks happens at two levels in MIC. Firstly, we (optionally) build a shared covariance matrix for the tasks (important when the various tasks are nearly identical), and secondly, we use a shared coding scheme to specify which of the tasks (models) each feature is added to. Among many attractive properties of the method is its immunity to overfitting and the absence of any free parameters that might require cross validation, unlike ℓ_1 regularization methods. Results on synthetic and real data demonstrate that MIC performs better than state-of-the-art Multi-Task Learning Methods, not only in terms of accuracy but also in terms of simplicity of the resulting models. MIC works best when there are many features of which only a few are relevant, a situation that occurs commonly in computational biology and bioinformatics applications.

References

1. Caruana, R.: Multitask learning. In: Machine Learning, pp. 41–75 (1997)
2. Ando, R., Zhang, T.: A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. The Journal of Machine Learning Research 6, 1817–1853 (2005)
3. Jacob, L., Bach, F., Vert, J.P.: Clustered multi-task learning: A convex formulation. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 21 (2009)
4. Argyriou, A., Evgeniou, T., Pontil, M.: Convex multi-task feature learning. Mach. Learn. 73(3), 243–272 (2008)
5. Ben-David, S., Borbely, R.S.: A notion of task relatedness yielding provable multiple-task learning guarantees. Mach. Learn. 73(3), 273–287 (2008)

6. Lee, S.I., Chatalbashev, V., Vickrey, D., Koller, D.: Learning a meta-level prior for feature relevance from multiple related tasks. In: ICML 2007: Proceedings of the 24th international conference on Machine learning, pp. 489–496. ACM, New York (2007)
7. Raina, R., Ng, A.Y., Koller, D.: Constructing informative priors using transfer learning. In: ICML 2006, pp. 713–720. ACM, New York (2006)
8. Jebara, T.: Multi-task feature and kernel selection for SVMs. In: ICML 2004, ACM Press, New York (2004)
9. Turlach, B., Venables, W., Wright, S.: Simultaneous variable selection. Technometrics 47(3), 349–363 (2005)
10. Obozinski, G., Taskar, B., Jordan, M.I.: Joint covariate selection and joint subspace selection for multiple classification problems. Statistics and Computing (2009)
11. Efron, B., Hastie, T., Johnstone, L., Tibshirani, R.: Least angle regression. Annals of Statistics 32, 407–499 (2004)
12. Natarajan, B.: Sparse approximate solutions to linear systems. SIAM journal on computing 24, 227 (1995)
13. Lin, D., Pitler, E., Foster, D.P., Ungar, L.H.: In defense of ℓ_0. In: Workhsop on Feature Selection at International Conference on Machine Learning, ICML 2008 (2008)
14. Rissanen, J.: Hypothesis selection and testing by the mdl principle. The Computer Journal 42, 260–269 (1999)
15. George, E., Foster, D.: Calibration and empirical Bayes variable selection. Biometrika 87(4), 731–747 (2000)
16. Foster, D.P., George, E.I.: The risk inflation criterion for multiple regression. The Annals of Statistics 22(4), 1947–1975 (1994)
17. Zhou, J., Foster, D., Stine, R., Ungar, L.: Streamwise feature selection. The Journal of Machine Learning Research 7, 1861–1885 (2006)
18. Rissanen, J.: A universal prior for integers and estimation by minimum description length. Annals of Statistics 11(2), 416–431 (1983)
19. Elias, P.: Universal codeword sets and representations of the integers. IEEE Transactions on Information Theory 21(2), 194–203 (1975)
20. Friedman, J.: Fast Sparse Regression and Classification (2008)
21. Litvin, O., Causton, H.C., Chen, B., Pe'er, D.: Special feature: Modularity and interactions in the genetics of gene expression. Proceedings of the National Academy of Sciences of the United States of America (February 2009) PMID: 19223586
22. van't Veer, L.J., Dai, H., van de Vijver, M.J., He, Y.D., Hart, A.A., Mao, M., Peterse, H.L., van der Kooy, K., Marton, M.J., Witteveen, A.T., Schreiber, G.J., Kerkhoven, R.M., Roberts, C., Linsley, P.S., Bernards, R., Friend, S.H.: Gene expression profiling predicts clinical outcome of breast cancer. Nature 415(6871), 530–536 (2002)
23. Kao, W.C., Rakhlin, A.: Transfer learning toolkit (2007), `http://multitask.cs.berkeley.edu`