Journal of Machine Learning Research 1 (2010) - Submitted 5/10; Published /

Minimum Description Length (MDL) Based Models for Sparse Learning

Paramveer S. Dhillon DHILLON@CIS.UPENN.EDU
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA

Dean Foster FOSTER@WHARTON.UPENN.EDU
Department of Statistics
Wharton School, University of Pennsylvania
Philadelphia, PA 19104, USA

Lyle H. Ungar UNGAR@CIS.UPENN.EDU
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA

Editor: Francis Bach

Abstract

We propose a framework for learning sparse models based on the information theoretic Minimum Description Length (MDL) principle. We present two models, MIC-SINGLE and MIC-MULTI, which use ℓ_0-penalties derived using two-part MDL codings to select features. MIC-SINGLE assumes that the features are divided into groups and induces a two level sparsity, selecting a subset of the feature groups, and also selecting features within each selected group. MIC-MULTI applies when there are multiple related tasks that share the same set of potentially predictive features. It also induces a two level sparsity, selecting a subset of the features, and then selecting which of the tasks each feature should be added to. Lastly, we propose a model, TRANSFEAT, that can be used to transfer knowledge from a set of previously learned tasks to a new task that is expected to share similar features. All three methods are designed for selecting a small set of predictive features from a large pool of candidate features. We demonstrate the effectiveness of our approach with experimental results on data from genomics and from word sense disambiguation problems.

1. Introduction

Classical supervised learning algorithms use a set of feature-label pairs to learn mappings from the features to the associated labels. They generally do this by considering each classification task (each possible label) in isolation and learning a model for that task. Learning models independently for different tasks often works well, but when the labeled data is limited and expensive to obtain, an attractive alternative is to build shared models for multiple related tasks (Caruana, 1997; Ando and Zhang, 2005). For example, when one is trying to predict a set of related responses ("tasks"), be they multiple clinical outcomes for patients or growth rates for yeast strains under different conditions, it may be possible to "borrow strength" by sharing information between the models for the different responses. Inductive

©2010 Paramveer S. Dhillon and Dean Foster and Lyle H. Ungar.

transfer by building shared models can also be valuable when we have a disproportionate amount of labeled data for "similar" tasks. In such a case, building separate models for each task often gives poor predictive accuracies on tasks which have little data.

As a running example, we consider the problem of disambiguating word senses based on their context. Here, each observation of a word (e.g., a sentence containing the word "fire") is associated with multiple labels corresponding to each of the different possible meanings (e.g., for firing a person, firing a gun, firing off a note, etc.). Given the high-dimensional nature of Word Sense Disambiguation (WSD) data, feature selection is important for both linguistic understanding and for effective prediction (Chen et al., 2006). Also, since the features that are useful for predicting one sense are likely to be useful for predicting the other senses (perhaps with a coefficient of different sign.), we propose to select features that are useful in predicting these multiple responses.

Another closely related problem is grouped feature selection; i.e., enforcing sparsity at the level of groups (feature classes) (Yuan and Lin, 2006; Bach et al., 2004; Dhillon et al., 2008). In this problem the group structure is over the features rather than over the tasks. Multi-task learning (described above) can also be thought of as a special case of this "group sparsity" scenario in which a group is defined by fixing a specific feature and ranging over multiple tasks. The block-norm approach to these problems uses a combination of ℓ_1 and ℓ_2 norms as regularization terms and adds each feature into the models of either none or all of the tasks (Obozinski et al., 2009) for the multi-task case and selects either none or all the features from a given group in the case of group sparsity. However, these approaches are known to overestimate the support (Wainwright, 2009) as they select more features than the correct set of sparse features that generated the data. (Wainwright, 2009; Liu and Zhang; Nardi and Rinaldo, 2008) have showed that certain scalings of the regularization coefficient yields more sparse solutions, which have with high probability the same support as the model generating the data. Even then there are further problems with these methods; in order to obtain very sparse solutions, one has to use a large regularization parameter that leads to suboptimal prediction accuracy because this high penalty not only shrinks irrelevant features to zero, but also shrinks relevant features to zero (Zhang, 2009a). Another alternative is to threshold the obtained coefficients (Lounici, 2008), but this introduces another thresholding parameter which needs to be tuned.

Motivated by the aforementioned reasons and by recent theoretical results on ℓ_0 penalty based regularization (Zhang, 2009a,b), we consider ℓ_0 penalty based formulations in this paper. In particular we propose to solve these two related problems, simultaneous feature selection for a set of multiple related tasks and grouped feature selection for a single task, by using coding schemes inspired by the Minimum Description Length (MDL) principle. We propose a common framework for these problems which we call the Multiple Inclusion Criterion (MIC). We use a "two part" version of MDL (Grünwald, 2005) to define a cost function which is greedily minimized by our methods. Since the greedy feature selection approximates the ℓ_0 penalty, we achieve a high degree of sparsity as is desired for both scientific interpretability and for accurate prediction in domains like Genomics and Word Sense Disambiguation (WSD) which have very high dimensional data. More importantly, our methods achieve *two-level sparsity*. In multi-task learning, each feature is added into models of a (possibly empty) subset of the tasks and in group feature selection, a (possibly empty) subset of the features are selected from each group (feature class).

We also propose a similarly motivated model (TransFeat) (Dhillon and Ungar, 2009) for "intra-domain" adaptation which can be used to transfer knowledge from a set of already learned tasks to a new task which is similar to the aforementioned tasks. As an example, consider the task of predicting whether a word has a given sense when one already has models for predicting synonyms of that word. These models are likely to share many of the same features; i.e., a model for disambiguating one sense of "discharge" is likely to use many of the same features as one for disambiguating the sense of "fire" which is its synonym. Unlike MIC where we do simultaneous feature selection, the sharing in this case takes the form of a prior. TransFeat is most beneficial when the word under consideration has considerably less labeled data available than the synonyms of that word (for example) so that building a supervised learning model for that word alone does not yield high predictive accuracy.

The rest of the paper is organized as follows. In the next section we provide background on feature selection and the MDL principle. In Section 3, we review relevant previous work. Then in Section 4 we develop the general framework used by our models and describe the MIC-Multi and MIC-Single models in detail. In Section 5, we show experimental results on real and synthetic data. In Section 6, we provide some model consistency results for the MIC models. In Section 7, we discuss the TransFeat model and show its effectiveness for intra-domain adaptation on real world datasets. We conclude in Section 8 with a brief summary.

2. Background

We assume a setting in which we are given n labeled data samples as $\{(x_i, y_i)_{i=1}^n \in \mathcal{X} \times \mathcal{Y}\}$ where $\mathcal{X} \in \mathbb{R}^p$ (the feature vector lives in a p dimensional space) and our goal is to find the parameter vector ($w \in \mathbb{R}^p$) of a statistical model fit to the above data. Alternatively, we can represent the data and the response variables in matrix form as $X_{n\times p}$ and $Y_{n\times 1}$, respectively, and the p dimensional weight vector as $w_{p\times 1}$. Standard linear or logistic regression models of the form $Y = w \cdot X$ (or $Y = \frac{1}{1+e^{-w\cdot X}}$) fail to estimate the weight vector w in the case in which $p > n$ as they require inversion of a rank deficient matrix. To overcome this problem, *regularized* versions of the linear or logistic regressions are used which penalize some norm of the weight vectors:

$$\hat{w} = \underset{w}{\operatorname{argmin}}\{||Y - X \cdot w||_2 + \lambda||w||_p\} \tag{1}$$

where $||w||_p$ represents the ℓ_p norm of w and λ is a hyperparameter.

For $p = 2$, the penalized regression is known as *Ridge Regression*, which corresponds to a Bayesian maximum a posteriori estimate for w under a Gaussian prior and shrinks the weight vector but does not enforce sparsity. The ℓ_1 penalty (Lasso) is equivalent to a double exponential prior on w (Tibshirani, 1996) and enforces sparsity by driving some of the weights to zero. As p approaches 0, $||w||_p$ approaches the number of non-zero values in w. Hence regularization with ℓ_0 penalty is subset selection: Choosing a small number of the original features to retain in the model. Once a coefficient is in the model, all that counts is the cost of adding it in the first place. The ℓ_0 penalty has a number of advantages, including bounded worst case risk with respect to the ℓ_1 penalty and better control of False Discovery Rate (FDR) (Lin et al., 2008). There are other problems with the ℓ_1 penalty other than being

less sparse as mentioned earlier, namely that its sparsity is not explicitly controlled, and in order to obtain very sparse solutions, one has to use a large regularization parameter that leads to suboptimal prediction accuracy because this high penalty not only shrinks irrelevant features to zero, but also shrinks relevant features to zero (Zhang, 2009a). However, one virtue of the ℓ_1 penalty is computational tractability (Efron et al., 2004), in contrast to the ℓ_0 penalty, which requires subset search which is (worst case) NP-Hard (Natarajan, 1995). In practice, approximate greedy algorithms like forward stepwise feature selection yield accurate, highly sparse solutions.

In a regression model, the residual sum of squares is proportional up to an additive constant to the negative log-likelihood of $\mathbf{Y}$ given $\mathbf{X}$ (Bickel and Doksum, 2001). Thus, the ℓ_0 regularization can be rephrased as a penalized likelihood criterion as follows:

$$score = -2\log(P(Y|\hat{w}_q)) + F \cdot q \tag{2}$$

where q is the number of features in the model, $P(Y|\hat{w}_q)$ is the likelihood of the data given a model containing q features and F is a free parameter that controls the amount of weight placed on the ℓ_0 norm. Various penalties have been proposed for F, including

- $F = 2$, corresponding approximately to the AIC (Akaike Information Criterion) (Akaike, 1973),
- $F = \log n$, giving the BIC (Bayesian Information Criterion) (Schwartz, 1978),
- $F = 2\log p$, giving to RIC (Risk Inflation Criterion—similar to a "Bonferroni correction") (Foster and George, 1994).

As discussed in next subsection, each of these penalties can also be derived by using the Minimum Description Length (MDL) principle under different coding schemes.

2.1 Minimum Description Length (MDL) Principle Preliminaries

MDL (Rissanen, 1978, 1999) is a principle for model selection which treats the best model as the one which maximally compresses a digital representation of the observed data. We can envision a "Sender" who wants to transmit some data to a "Receiver" using as few bits as possible. For an illustrative example of the MDL principle, consider the case of simple linear regression. Assume that both the Sender and Receiver know the $n \times p$ data matrix X, and the Sender wants to convey the values in the $n \times 1$ response matrix Y. The naïve way to do this would be to send the raw values for each of the n observations of Y. However, a more efficient way to send this information would be to describe a regression model $\hat{w}$ for Y given X and then to send the residuals $Y - X \cdot \hat{w}$, which have a much narrower distribution and would require fewer bits to encode.

To minimize description length, then, Sender should choose $\hat{w}^*$ such that

$$\hat{w}^* = \underset{\hat{w}}{\operatorname{argmin}}\{\mathcal{D}(Y|\hat{w}) + \mathcal{D}(\hat{w})\}, \tag{3}$$

where the first term is the description length of the residuals about the model, and the second term is the description length of the model itself. In other words, the first term represents the fit of the model to data; as the model fits better this term shrinks. The

second term represents the complexity of the model; it grows as the model becomes more complex.

This version of the MDL principle is known as "*Two part* MDL" (Grünwald, 2005); the exact meaning of both these terms is described in the following sections.

2.1.1 Coding the Data: $\mathcal{D}(Y|\hat{w})$

The Kraft inequality in information theory (Cover and Thomas, 2006) implies that for any probability distribution $\{p_i\}$ over a finite or countable set, there exists a corresponding code with codeword length $\lceil -\lg p_i \rceil$. Moreover, these code lengths are optimal in the sense of minimizing the expected code length with respect to $\{p_i\}$. Also, if the Sender and Receiver agree on a model (e.g. linear regression), then they have a probability distribution over the residuals ϵ, so they will agree to use a code for the residuals with length:

$$\mathcal{D}(Y|\hat{w}) = -\lg P(\epsilon|\hat{w}) = -\lg P(Y|\hat{w}) \tag{4}$$

i.e., the negative log-likelihood of the data given the model. We dropped the ceiling on $-\lg P(Y|\hat{w})$ since we use "idealized" code lengths (Barron et al., 1998).

Consider a forward stepwise-regression setting in which we have already added $q-1$ features to our model (including the intercept term), and we are deciding whether to include an extra q^{th} feature. Let Y_i denote the i^{th} row of Y and $\hat{w}_q$, a linear regression model with all q features, then:

$$\begin{aligned}
\mathcal{D}(Y|\hat{w}) &= -\lg \prod_{i=1}^{n} P(Y_i|\hat{w}_q) \qquad (5)\\
&= -\sum_{i=1}^{n} \lg \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(Y_i - X_i \cdot \hat{w}_q)^2 \right) \right] \\
&= \frac{1}{2\ln 2}\left[n\ln(2\pi\sigma^2) + \frac{(\mathbf{Y} - \mathbf{X}\cdot\hat{w}_q)^2}{\sigma^2} \right],
\end{aligned}$$

σ^2 is unknown in practice, but it can be estimated[1] as:

$$\hat{\sigma}^2 = \frac{(\mathbf{Y} - \mathbf{X}\cdot\hat{w}_{q-1})^2}{n} \tag{6}$$

We can write the final expression for $\mathcal{D}(Y|\hat{w})$, incorporating $\hat{\sigma}^2$ as:

$$\mathcal{D}(Y|\hat{w}) = \frac{n}{2\ln 2}\left[\ln\left(\frac{2\pi \times (\mathbf{Y} - \mathbf{X}\cdot\hat{w}_{q-1})^2}{n} \right) + \left(\frac{\mathbf{Y} - \mathbf{X}\cdot\hat{w}_q}{\mathbf{Y} - \mathbf{X}\cdot\hat{w}_{q-1}} \right)^2 \right]. \tag{7}$$

1. This is the ML (Maximum Likelihood) estimate for σ^2 which Sender uses, as ignoring the model-coding cost, maximizing likelihood is equivalent to minimizing description length. Some statisticians, in practice, use the unbiased estimate $\hat{\sigma}^2 = \frac{(\mathbf{Y}-\mathbf{X}\cdot\hat{w}_{q-1})^2}{n-q}$.
In the experiments presented in this paper, we estimate $\hat{\sigma}^2$ without the current q^{th} feature in model, in order to prevent overfitting.

2.1.2 Coding the Model: $\mathcal{D}(\hat{w})$

Just as $\mathcal{D}(Y|\hat{w})$ depends on the model for the residuals that Sender and Receiver choose, so their coding scheme for $\hat{w}$ itself will reflect their prior expectations[2]. When the number of features p is large (say, > 1000), Sender will likely only want to transmit a few of them that are most relevant, and hence the $\hat{w}$ will contain mostly zeros. So, the first step in coding $\hat{w}$ could be to say where the non-zero entries are located; if only a few features enter the model, this can be done relatively efficiently by listing the indices of the features in the set $\{1, 2, ..., p\}$. This requires $\lceil \lg p \rceil$ bits or approximately $\lg p$ bits.

The second step is to encode the numerical values of those coefficients. (Rissanen, 1983) suggested the basic approach for doing this by creating a discrete grid over some possible parameter values, and use a code for integers to specify which grid point is closest. A simple way to approximate the value of a particular coefficient $\hat{w}$ is to encode an integer version of its z-score relative to the null-hypothesis value w_0 (which in our case is 0):

$$\left\langle \frac{\hat{w} - w_0}{SE(\hat{w})} \right\rangle = \left\langle \frac{\hat{w}}{SE(\hat{w})} \right\rangle, \tag{8}$$

where $\langle x \rangle$ means the closest integer to x and SE represents standard error. The z-score can be coded with the idealized universal code for the positive integers of (Rissanen, 1983), in which the cost to code $i \in 1, 2, 3, ...$ is

$$\lg^* i + b,$$

where $\lg^* i = \lg i + \lg \lg i + \lg \lg \lg i + \ldots$ so long as the terms remain positive, and $b \approx \lg 2.865 \approx 1.516$ is the constant such that

$$\sum_{i=1}^{\infty} 2^{-(\lg^* i + b)} = 1$$

We require the $\lg^*$ instead of a simple lg because the number of bits Sender uses to convey the integer i will vary, and she needs to tell the Receiver how many bits to expect. The number of bits is itself an integer than can be coded, hence the iteration of logarithms.

In fact, in practice it is unnecessary to allow our integer code to extend to arbitrarily large integers. We are interested in features near the limit of detectability and we expect our z-scores to be roughly in the range ~ 2 to ~ 4, since if they were much higher, the true features would be obvious and would not require sensitive feature selection. We could thus impose some maximum possible z-score Z that we might ever want to encode (say, 1000) and assume that all of our z-scores will fall below it. In this case, the constant c can be reduced to a new value c_Z, now only being large enough that,

$$\sum_{i=1}^{Z} 2^{(-\lg^* i + c_Z)} = 1 \tag{9}$$

2. By the Kraft inequality, we can interpret $2^{-\mathcal{D}(\hat{w})}$ as a prior over possible models w. In fact, this is done explicitly in the Minimum Message Length (MML) principle which is a Bayesian analogue of MDL, which chooses the model $\hat{w}$ with maximum $P(w|Y)$ i.e., it chooses the model that minimizes $-\lg P(w|Y) = -\lg P(w) - \lg P(Y|w) + const.$

In particular $c_{1000} \approx 1.199$. In our implementation in this paper, we avoid computing the actual values of our z-scores and instead assume a constant 2 bits per coefficient. Though MDL theoretically has no tunable parameters once Sender and Receiver have decided the models, the number of bits to specify a coefficient can act as one. We found 2 bits per coefficient to work well in practice.

Combining the cost of the residuals with the cost of the model gives the following formula for the description length as a function of number of features that we include in the model:

$$-\lg P(Y|\hat{w}) + q(\lg p + 2) \tag{10}$$

where q is the number of features in the model and p is the total number of candidate features.

The above formula represents the simplest possible coding scenario and we will refer to it later in the paper as "**Baseline Coding Scheme**" when we propose more complex coding schemes for the problems of simultaneous feature selection for a set of multiple related tasks and grouped feature selection for a single task.

3. Related Work

The main contribution of this paper is to propose a joint framework for the related tasks of simultaneous feature selection for multiple related tasks and grouped feature selection for a single task. We are not aware of any previous work that addresses these two problems together, though (Obozinski et al., 2009) do mention that these two problems are related. Nonetheless, there has been much previous work on each of these problems separately.

(Jebara, 2004) use 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). (Turlach et al., 2005) consider the case $p = \infty$, while (Argyriou et al., 2008; Obozinski et al., 2009) use $p = 2$. (Argyriou et al., 2008) show that the general subspace selection problem can be formulated as an optimization problem involving the trace norm. (Obozinski et al., 2009) propose BBLasso, which focuses on the case where the trace norm is not required; they instead use a homotopy-based approach to evaluate the entire regularization path efficiently (Efron et al., 2004). (Ando and Zhang, 2005) also propose a framework which uses multiple prediction problems to learn an underlying shared structural parameter on the input (feature) space and they penalize the weight vectors by ℓ_2 norm. 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.

The related problem of grouped feature selection for a single task has also been addressed previously by (Yuan and Lin, 2006; Bach et al., 2004; Meier et al., 2008; Zhao et al., 2008) and is known as "Group Lasso". It is an extension of Lasso (ℓ_1 penalty) to the case of grouped structure in data and it enforces sparsity at the level of groups i.e. an entire group of features is selected. It penalizes a (ℓ_1/ℓ_2) norm of the feature weights. An alternative

formulation of Group Lasso is called Multiple Kernel Learning (MKL) (Bach et al., 2004; Bach, 2008); it penalizes the kernel Hilbert norm instead of the Euclidean norm.

Our approach is different from these methods in that we use ℓ_0 penalty-based greedy feature selection methods which minimize a cost function provided by MDL based coding schemes. MDL-based coding schemes provide much flexibility to incorporate arbitrary sparsity structures in the problem at hand. Recently (Huang et al., 2009) have also used coding schemes similar to the MDL for enforcing arbitrary structured sparsity patterns over the feature space.

4. Multiple Inclusion Criterion (MIC)

MIC is a general framework for ℓ_0 penalty based greedy feature selection which minimizes a cost function provided by the Minimum Description Length (MDL) principle. MIC provides an elegant way of incorporating arbitrary sparsity patterns in the feature space by using MDL coding schemes customized to the problem at hand. In this section, we describe how MIC can be used to provide statistically efficient models for the problems of simultaneous feature selection for multiple related tasks and grouped feature selection for a single task. To do that, we first introduce some more notation and follow up on the MDL introduction in Section 2.1.

For the problem of simultaneous feature selection for a set of related tasks (which is addressed using MIC-Multi) we assume a set of h regression or classification tasks which can potentially share a set of p features and a total of n labeled training examples. The task is to learn a set of joint ("shared") models for all the h tasks. We represent the feature, response and the weight matrices as $\mathbf{X}_{n\times p}$, $\mathbf{Y}_{n\times h}$ and $\mathbf{w}_{p\times h}$ respectively. Additionally, for simplicity of analysis we assume a linear regression setting of the form [3] $Y = X \cdot w + \epsilon$ with a Gaussian noise term $\epsilon_{n\times h}$. Note that the noise on the responses (ϵ) may be correlated; for instance, if our responses consist of temperature measurements at various locations, taken with the same thermometer, then if our thermometer drifted high at one location, it will have been high at the other location also. Thus, we take the rows of ϵ to have non-zero covariance:

$$\epsilon_i \sim \mathcal{N}_h(0, \Sigma), \tag{11}$$

where ϵ_i is the i^{th} row of ϵ and Σ is an arbitrary $h \times h$ covariance matrix.

Similarly, for the related problem of grouped feature selection (which is addressed using MIC-Single) also, we have a total of p candidate features which are further divided into K groups (equal or unequal). Again, we assume the availability of a (fixed number) n of labeled training examples. Just as above we can represent the feature, response and weight matrices as $\mathbf{X}_{n\times p}$, $\mathbf{Y}_{n\times 1}$ and $\mathbf{w}_{p\times 1}$ respectively.

Let S represent the total description length (TDL) of the MDL message that is exchanged between the Sender and the Receiver. In the case of MIC-Multi, S is the combined message length for all h tasks and hence we select features for all the h tasks simultaneously to minimize S and in the case of MIC-Single it can either be the combined message length

3. It can be extended to the standard classification setting by replacing the squared loss with a logistic loss, but due to lack of closed form solutions for logistic regression and since correlation between residuals is inconvenient to model in classification settings, we refrain from analysing them.

for all the features within a given group (feature class) (MIC-Single-(I)) or the message length of a given feature (MIC-Single-(II)). Thus, when we evaluate a feature for addition into the model, we want to maximize the reduction of TDL by adding that feature to our model. More formally, at each iteration we greedily add those features to our model that:

$$\begin{aligned} \Delta S^i &= \Delta S^i_E - \Delta S^i_M, \\ \textit{Best Feature} &= \underset{\hat{i}}{\text{argmax}}\{\Delta S^i\} \end{aligned} \tag{12}$$

where $\Delta S_E \geq 0$ is the reduction in residual-error coding cost i.e. the first term on right hand side in Equation 3, due to the increase in data likelihood given this new feature and $\Delta S_M > 0$ is the increase in model cost to encode the new feature (second term in Equation 3) and i ranges over all the p features.

In the next subsections we describe how we code the S_E and S_M terms (i.e. the residual error and model) for MIC-Multi and MIC-Single in detail.

4.1 MIC-Multi

As mentioned earlier, MIC-Multi borrows strength across multiple tasks and hence selects a joint set of features for related tasks (Dhillon et al., 2009).

4.1.1 Coding the Model

MIC-Multi borrows strength across responses by efficiently specifying the feature-response pairs in the $p \times h$ matrix $\hat{w}$. The naïve approach would be to put each of the ph coefficients in a linear order and specify the index of the desired coefficient using $\lg(mh)$ bits. But we can do better. If we expect nearly all the responses to be correlated with the predictive features, we could give all the responses nonzero coefficients (using $2h$ bits to code each of the h response coefficients and simply specify the feature that we are talking about by using $\lg p$ bits, as in Section 2.1.2. From now on we will refer to this approach as Full-MIC-Multi (*fully dependent MIC*-Multi) coding scheme, as it assumes that a selected feature will be added in the models of all the tasks, in much the same way as BBLasso (Obozinski et al., 2009). Another limiting case is the one when we do feature selection for all the tasks independently (the baseline "Independent" Coding Scheme); the coding scheme in that case takes the form given in Equation 10.

However, these assumptions are usually unrealistic; each feature is generally neither correlated with almost all the responses nor with none of the responses, but is rather correlated with a few of them. A more flexible coding scheme would allow us to specify only the subset of the responses to which we want to give nonzero coefficients. For instance, suppose we are considering feature number 2609; and, of the $h = 20$ responses, we think that only {3, 7, 14, 17 } should have nonzero coefficients with the current feature. Then, we can use $\lg p$ bits to specify our feature (number 2609) once, and then we can list the particular responses that have nonzero coefficients with feature 2609, thereby avoiding paying the cost of $\lg(mh)$ four times to specify each coefficient in isolation.

A standard practice in information theory literature to code a subset of size h is to first specify how many $k \leq h$ elements the subset contains and then which of the $\binom{h}{k}$ possible subsets with k elements we are referring to (Cover and Thomas, 2006). In particular, we

choose to code k using $\lg^* k + c_h$ bits, with c_h as defined in Equation 9. We then need $\lg \binom{h}{k}$ additional bits to specify the particular subset. We refer to this code as *partially dependent* MIC-Multi or simply Partial-MIC-Multi.

The total cost (S_M^i) to code the model of a feature for MIC-Multi is composed of three parts as follows:

$$S_M^i = \ell_H + \ell_I + \ell_\theta, \tag{13}$$

where ℓ_H is the number of bits needed to specify the subset k of the h tasks models in which to include the feature; ℓ_I is the number of bits used to describe which feature is being added and ℓ_θ is the description length of the coefficients of non-zero features.

We have already described the cost for ℓ_H above; it is equal to:

$$\ell_H = \lg^* k + c_h + \lg \binom{h}{k} \tag{14}$$

For ℓ_θ, we use a cost of 2 bits per coefficient, the motivation for which was described earlier in Section 2.1.2. For ℓ_I, which specifies the size of the code for the given feature, we use $\lg p$ bits, which is equivalent to a uniform prior over the features [4] i.e., each feature is equally likely to be selected. This can be accomplished by simply keeping a linear array of features and coding the indices of the features with nonzero coefficients.

Thus, we can represent the total model cost for MIC-Multi as:

$$S_M = \left(\lg^* k + c_h + \lg \binom{h}{k} \right) + (\lg p) + (2k) \tag{15}$$

4.1.2 Coding the Data

Let $\mathbf{E}$ be the residual error ($\mathbf{Y}$ - $\mathbf{X}\cdot\mathbf{w}$) matrix, and as mentioned above, let ϵ_i, $i = 1, 2, \ldots, n$ denote the i^{th} row of the error and let Σ be its $h \times h$ covariance matrix. The model likelihood under the Gaussian assumption[5] can be written as:

$$P(Y_i|\hat{w}_q) = \frac{1}{\sqrt{(2\pi)^h |\Sigma|}} \exp\left(\frac{1}{2}\epsilon_i^T \Sigma^{-1} \epsilon_i\right) \tag{16}$$

$$\begin{aligned} S_E &= -\lg \prod_{i=1}^{n} P(Y_i|\hat{w}_q) \\ &= \frac{1}{2\ln 2}\left[n \ln\left((2\pi)^h |\Sigma|\right) + \sum_{i=1}^{n} (Y_i - X_i \cdot \hat{w}_q)^T \Sigma^{-1} (Y_i - X_i \cdot \hat{w}_q) \right] \end{aligned} \tag{17}$$

with subscript i denoting the i^{th} row. Since Σ is in fact unknown, we estimate it using maximum likelihood (ML):

$$\hat{\Sigma}_F = \frac{1}{n} \left(\mathbf{Y} - \mathbf{X} \cdot \hat{w}_{q-1}\right)^T \left(\mathbf{Y} - \mathbf{X} \cdot \hat{w}_{q-1}\right), \tag{18}$$

4. The uniform code gives the worst-case minimax optimal code lengths (Grünwald, 2005).
5. As mentioned earlier, we are considering linear regression for simplicity of analysis and ease of modeling the correlation between residuals.

where the subscript F stands for "full covariance", and we use $\hat{w}_{q-1}$ to get ML estimate, instead of $\hat{w}_q$ to prevent overfitting, as we mentioned in Section 2.1.1.

In practice, we however find that estimating all the h^2 entries of the covariance matrix can lead to overfitting. Therefore we use shrunken estimates of the form $\hat{\Sigma}_\lambda = \lambda\hat{\Sigma}_D + (1-\lambda)\hat{\Sigma}_F$ for ($\lambda \in [0,1]$), which tend to work well in practice. We describe more technical details about our implementation in the Experiments section.

4.1.3 Comparison of various MIC-Multi Coding Schemes

In this section we discussed three MDL based information-theoretic approaches to multitask feature selection, namely Full-MIC-Multi, Baseline "Independent" Coding Scheme and Partial-MIC-Multi. In general, the negative log-likelihood portion of *Independent* may differ from that of the other two methods, because *Full* and *Partial* can use a non-diagonal covariance estimate like $\hat{\Sigma}_F$ or $\hat{\Sigma}_\lambda$, while *Independent* only operates on one response at a time, and thus implicitly uses $\hat{\Sigma}_D$. However, since we generally use $\hat{\Sigma}_\lambda$, due to the reasons mentioned earlier, for *Full* and *Partial*, the real difference comes from the coding schemes.

The coding costs for these three methods are compared in Table 1 for $p = 2000$ features, $h = 20$ responses, and for various values of k, the number of responses to which we add the current feature under consideration. Full-MIC-Multi is only allowed to take $k = 0$ or $k = h$, so it has h nonzero coefficients in all three rows of the table. However, if the extra $h - k$ coefficients correspond to non-predictive features, the extra reduction in residual-coding cost that Full-MIC-Multi enjoys over the other methods is likely to be small. As expected, each coding scheme is cheapest in the case for which it was designed; however, the MIC-Multi methods are never excessively expensive, unlike *Independent* for $k = h$.

Table 1: 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. $p \gg h \gg 1$, $\ell_I = \lg p$, $\ell_{\hat{\theta}} = 2$, and for $h \in \{5, \ldots, 1000\}$, $c_h \approx 1$. Examples of these values for $p = 2{,}000$ and $h = 20$ appear in brackets; the smallest of the costs appears in bold. **Note:** The costs are given per feature.

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

4.2 MIC-Single

MIC-Single is the algorithm for grouped feature selection, when features fall into *groups* or *classes* (Dhillon et al., 2008; Yuan and Lin, 2006; Bach et al., 2004). For example, genes can be divided into gene classes based on what pathway they occur in or features of a word can be

grouped based on whether they are based on specific neighbouring words, parts of speech, or more global document properties. More generically, starting from any set of features, one can generate new classes of features by using projections such as principle components analysis (PCA) or non-negative matrix factorization (NNMF), transformations such as log or square root, and interactions (products of features) (Dhillon et al., 2010). The problem of grouped feature selection (MIC-Single) is very closely related to the problem of simultaneous feature selection for a set of related tasks (MIC-Multi) as has also been pointed out by (Obozinski et al., 2009). The multi-task problem we described earlier can also be thought of as a grouped feature selection scenario in which a group is defined by fixing a specific feature and ranging over multiple tasks. Our MIC based models for these two problems also follow the same intuition; in (MIC-Multi) the tendency is to add a given feature into models of more and more tasks[6] and similarly in (MIC-Single) the tendency is to add more and more features from the same group as the whole rationale behind doing grouped feature selection is based on the fact that some feature groups contain highly predictive features than others.

4.2.1 Coding Schemes for MIC-Single

Since the problem of grouped feature selection is similar to the problem of simultaneous feature selection for a set of related tasks, we can propose a coding scheme which is analogous to the coding scheme for MIC-Multi. For example, in this case also we can code the data as $P(Y_i|\hat{w}_q) = \frac{1}{\sqrt{(2\pi)^{h_{single}}|\Sigma|}} \exp\left(\frac{1}{2}\epsilon_i^T \Sigma^{-1} \epsilon_i\right)$ in a similar fashion as Equation 16 where h_{single} is the number of features in a given group (feature class) and we will estimate the covariance matrix, which represents covariance between different features in the same group (feature class), in a similar way as we did for MIC-Multi i.e. by Maximum Likelihood Estimation. Remember, that in this case S_E term will be the message length for all the features within a given feature class. In a similar fashion, the number of bits to code the model can be represented as $S_M = \left[\lg^* k + c_h + \lg\binom{h_{single}}{k}\right] + \log p + 2k$, which corresponds to Equation 15[7]. The other mechanics of the coding scheme will also be the same as for MIC-Multi as this time we are trying to find a best subset of size k in a group (feature class) of size h_{single} and so we do a stepwise greedy search as earlier. From now on we refer to this coding scheme as MIC-Single (I).

Although this coding scheme works very well in practice, but it turns out that we are not exploiting the full flexibility that MDL based coding offer us. So, we propose a new coding scheme, which is computationally more efficient than MIC-Single (I), as it does not require a stepwise search for subset selection, though the predictive accuracy of both these coding schemes is comparable. We call this new computationally efficient coding scheme as MIC-Single (II) and it is explained in detail below.

Coding the data with MIC-Single (II): In this new coding scheme S_E is the message length for a single feature and ΔS_E represents the increase in likelihood of the data by adding that feature to the model.

6. The $\lg(\binom{h}{k})$ part of the model cost is only small when k is small or it is very large i.e. $k \approx h$ as $\binom{h}{k} = \binom{h}{h-k}$.
7. For simplicity of analysis and ease of comparison with coding schemes for MIC-Multi we are assuming that all groups are of the same size h_{single}, though in reality the groups may be of unequal size and the same coding scheme still holds.

Let $\mathbf{E}$ be the residual error ($\mathbf{Y}$ - $\mathbf{X}\cdot\mathbf{w}$) matrix as earlier, and let ϵ_i, $i = 1, 2, \ldots, n$ denote the i^{th} row of the error and let σ be variance of the Gaussian noise. The model likelihood can be written as:

$$P(Y_i|\hat{w}_q) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{\epsilon_i^2}{2\sigma^2}) \tag{19}$$

$$S_E = -\lg \prod_{i=1}^{n} P(Y_i|\hat{w}_q) \tag{20}$$

This equation is similar to the corresponding equation for MIC-Multi except that here we only have a single response (task). In this case also, the variance σ^2 is estimated using the Maximum Likelihood principle.

Coding the model with MIC-Single (II): This is where we differ from MIC-Single (I) and we use a coding scheme better suited to the group structure of the features. The intuition behind this coding scheme is that once we have selected (at least) one feature from a given group then it should become easier to select more features from the group or in other words, the cost of adding more features from the same feature class should be cheap. The total cost of the model can be seen as composed of three parts as:

$$S_M^i = \ell_C + \ell_I + \ell_\theta \tag{21}$$

where ℓ_C is the number of bits used to code the index of the group of the evaluated feature, ℓ_I is the number of bits to code the index of the evaluated feature (within that particular group) and ℓ_θ is the cost to code the coefficient of the evaluated feature.

Our coding scheme assumes a uniform prior over all the groups i.e. each group is equally likely to contain beneficial features[8] to begin with. So ℓ_C is $\lg K$ where K is the total number of groups (feature classes) in the data. Now, if a feature gets selected from a group (feature class) from which we had previously selected features, then we can save some bits by using "switch" coding and coding ℓ_C using only $1 + \lg Q$ bits where Q is the total number of groups (feature classes) included in the model till that point of time and 1 bit is used to represent that this group (feature class) has previously produced beneficial features. (Think of keeping an indexed list of length Q of the feature classes that have been selected). This is where our method wins over other methods and we do not need to code the same feature class over and over again if it has produced beneficial features in the past. Therefore ℓ_C is:

$$\ell_C = \begin{cases} \lg(K) & if\ the\ feature\ class\ is\ not\ in\ the\ model \\ 1 + \lg(Q) & if\ the\ feature\ class\ is\ already\ in\ the\ model \end{cases} \tag{22}$$

To code ℓ_I we again assume a uniform prior over all the features within that particular group. This corresponds to $\lg m_i$ bits where m_i is the total number of features in the feature class of which the i^{th} feature is a part of. This is pretty similar to RIC (Risk Inflation Criterion)

8. This is actually a pretty good assumption as mentioned earlier. The uniform code gives the worst-case minimax optimal code lengths (Grünwald, 2005) and hence it is reasonable to use it if the data distribution is *completely unknown* or if *no* distribution is assumed.

style (Foster and George, 1994) coding or the widely use Bonferroni penalty. Finally, to code ℓ_θ we use 2 bits per coefficient, the motivation for which was as described earlier. Therefore, the model cost per feature can be represented as:

$$S_M^i = ((\lg K)\ OR\ (1 + \lg Q)) + (\lg m_i) + 2 \tag{23}$$

As mentioned earlier, this coding scheme is computationally cheaper than MIC-Single (I) as it does not require a subset search every time a feature is added to the model and it provides comparable predictive accuracy to MIC-Single (I). Note that just analogous to MIC-Multi it is possible to come up with a new coding scheme called Full MIC-Single which just like its MIC-Multi counterpart would add all the features from a given group (feature class) into the model. The MIC-Single schemes presented here are the most general setting and are analogous to Partial MIC-Multi for the multi response (task) scenario.

4.3 Algorithms and Implementation Details

In this subsection we outline the algorithms for MIC-Multi and MIC-Single and also explain some details of the search strategy that we used for efficient subset search in case of MIC-Multi.

4.3.1 Algorithms

The algorithm for MIC-Multi is as described in Algorithm 1. We provide algorithm for the most general case i.e. Partial MIC-Multi as the other two cases i.e. *Full* and *Independent* are the special cases of this scenario.

The algorithm for MIC-Single (II) is described in Algorithm 2. The algorithm makes multiple passes through data and at each iteration adds the best feature to the model. It stops when no feature provides better ΔS than in the previous iteration. Since, it can be the case that it is not worth adding a single feature from a particular group (feature class) but it is still beneficial to add multiple features from that class. So, a clever search strategy that we found helpful with MIC-Single (II) was to use a mixed forward-backward greedy stepwise strategy in which one continues the search past the stopping criterion given in the algorithm and then sequentially removes the "worst" features from the now overfit model by making a "Backward" pass. In practice, we found this search strategy helpful. A similar hybrid forward-backward strategy was also used by (Zhang, 2009a).

Note that we do not provide algorithm for MIC-Single (I) as it is pretty similar to MIC-Multi with minor notational modifications as mentioned in the previous subsection.

4.3.2 Stepwise Search Method

Since MIC-Multi requires subset search over the set of possible tasks in which to consider a feature for addition, so a discussion of our greedy search strategy is warranted.

For each feature, we evaluate the change in TDL (Total Description Length) 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 $\mathcal{O}(pp_s)$, where p_s is the number of features eventually added.

Algorithm 1 Partial MIC-Multi

Include the intercept (feature number 1) in all h response models.
$remaining\_features = \{2, \ldots, p\}$.
$keep\_adding\_features =$ true.
while $keep\_adding\_features$ **do**
  for j in $remaining\_features$ **do**
    // Find the best subset of response models to which to add feature j.
    for $k = 1$ to h **do**
      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.)
      Compute ΔS^k_{jE}, the decrease in data residual cost, and ΔS^k_{jM}, the resulting increase in model-coding cost, relative to not including feature j in any response models.
    end for
    Let k_j be the value of k that maximizes $\Delta S^k_{jE} - \Delta S^k_{jM}$.
    $\Delta S_j := \Delta S^{k_j}_{jE} - \Delta S^{k_j}_{jM}$.
  end for
  Let j^* be the feature j that maximizes ΔS_j, the reduction in TDL for adding feature j.
  if $\Delta S_{j^*} > 0$ **then**
    Add feature j^* to the appropriate k_{j^*} response models.
    $remaining\_features = remaining\_features - \{j^*\}$.
  else
    $keep\_adding\_features =$ false.
  end if
end while

Algorithm 2 MIC-Single (II)

$flag = True$; // flag for indicating when to stop
$model = \{\}$; // initially no features in model
$prev\_max = 0$; // keeps track of the value of ΔS_E in the previous iteration
while {flag == True} **do**
  for {i = 1 to p} **do**
    Compute ΔS^i_E; // Increase in likelihood by adding feature 'i' to the model
    Compute ΔS^i_M; // Number of extra bits required to code the i^{th} feature
    $\Delta S^i := \Delta S^i_E - \Delta S^i_M$;
  end for
  $i_{max} := argmax_i\{\Delta S^i\}$; //The best feature in the current iteration
  $current\_max := max_i\{\Delta S^i\}$; //The best penalized likelihood change in the current iteration
  if $\{current\_max > prev\_max\}$ **then**
    $model := model \bigcup \{i_{max}\}$; // Add the current feature to model
    $prev\_max := current\_max$;
  else
    $flag := False$;
  end if
end while

To select the optimal number k of task models 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[9]. 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-Multi's coding cost does not increase monotonically with k, so even if adding the feature to an intermediate number of 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 $\mathcal{O}(h^2 p p_s)$ 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 $\mathcal{O}(p)$ sweep through the features to compute an upper bound on the decrease in TDL that could result from adding that feature as

$$\text{(decrease in TDL if the feature is added to all } h \text{ response models)} - \log p. \tag{24}$$

   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

$$\begin{aligned}&\text{(decrease in TDL if the feature is added to all } h \text{ response models)}\\ &- \left(\log^* k + c_k + \log \binom{h}{k} + 2k\right),\end{aligned} \tag{25}$$

   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[10].

5. Experimental Results

In this section we empirically show the usefulness of our MIC based models (MIC [Multi and Single]) on a variety of real world datasets pertaining to Genomics and Computational

9. 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 non-diagonal, 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 5, then an $\mathcal{O}(h)$ search through the tasks without re-evaluation suffices.
10. We say "no higher value of k" rather than "the next higher value of k" because (25) does not decrease monotonically with k, due to the $\log \binom{h}{k}$ quantity.

Linguistics (particularly Word Sense Disambiguation) domains. Besides this we also show results on synthetic datasets to illustrate the cases when our models are most beneficial.

5.1 MIC-Multi

In this section, we first evaluate the MIC-Multi approach on three synthetic datasets, each of which is designed to match the assumptions of, respectively, the Partial and Full MIC-Multi, and Baseline (Independent) coding scheme (Equation 10). We then test the methods on two biological data sets, a Yeast Growth dataset (Perlstein et al., 2007), which consists of real-valued growth measurements of multiple strains of yeast under different drug conditions, and a Breast Cancer dataset (van 't Veer, 2002), 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 4.1.3 against two other multitask algorithms: AndoZhang (Ando and Zhang, 2005) and BBLasso (Obozinski et al., 2009), as implemented in the Berkeley Transfer Learning Toolkit (Rakhlin, 2007). We did not compare MIC-Multi with other methods from the toolkit as they all require the data to have additional structure, such as *meta-features* (Lee et al., 2007; Raina et al., 2006), 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 (Ando and Zhang, 2005) 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 (Ando and Zhang, 2005).

MIC-Multi, as presented in Section 4.1.2, 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-Multi with a regular regression likelihood term. Once the features were selected, however, we used logistic regression applied to just those features to obtain MIC-Multi's actual model coefficients.

As noted in Section 4.1.2, MIC-Multi's negative log-likelihood term can be computed with an arbitrary $h \times h$ covariance matrix Σ among the h tasks. We did not estimate all the h^2 entries of Σ as it lead to overfitting, so we instead took Σ to be diagonal[11].

5.1.1 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

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

where $p = 2000$ 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

11. Informal experiments showed that estimating Σ as a convex combination of the full and diagonal estimates (i.e. $\hat{\Sigma}_\lambda$) also works well but we chose to use diagonal Σ (i.e. $\hat{\Sigma}_D$) due to its simplicity and to show the advantage of using a better coding scheme to code the model as by using diagonal Σ Partial and Independent methods are the same except S_M (i.e. cost of coding the model).

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 $p^* = 4$ 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 $(5, 6, \ldots, 2000)$.
- 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 candidate features $1, \ldots, 2000$.

For the synthetic data, we report precision and recall to measure the quality of feature selection. This can be done both at a coefficient[12] 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-Multi and BBLasso always make entire rows of their estimated w matrices nonzero and so tend to have larger numbers of nonzero coefficients. Table 2 shows the performance of each of the methods on five instances of the Partial, Full, and Independent synthetic data sets. On the *Partial* data set, Partial MIC-Multi performed the best, closely followed by Baseline (Independent); on the *Full* synthetic data, Full MIC-Multi and Partial MIC-Multi performed equally well; and on the *Independent* synthetic data, the *Baseline* algorithm performed the best closely followed by Partial MIC-Multi. 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 4.1.3.

The table shows that only in one of the three cases does non-MIC 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.

5.1.2 Evaluation on Real Datasets

This section compares the performance of MIC-Multi methods with AndoZhang and BBLasso on a Yeast and a Breast Cancer dataset. These are typical biological datasets in that only a handful of features are predictive from thousands of potential features. This is precisely the case in which MIC-Multi outperforms other methods. MIC-Multi not only gives better accuracy, but does so by choosing fewer features than BBLasso's $\ell_1 - \ell_2$-based approach.

12. A coefficient is defined as the addition of a given feature to a single task. For example if a feature was added to models of 10 tasks, then 1 feature and 10 coefficients were selected.

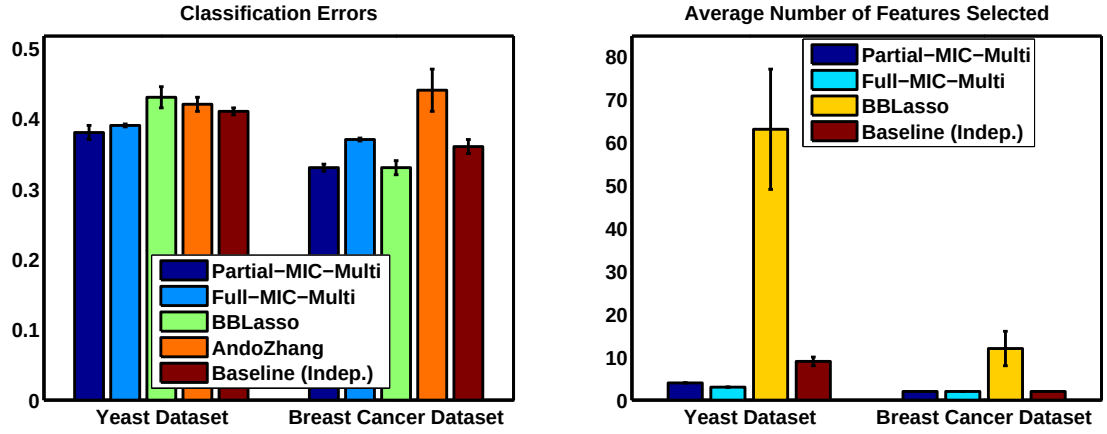
Table 2: Test-set accuracy, precision, and recall of MIC-Multi and other methods on 5 instances of various synthetic data sets generated as described in Section 5.1.1. Standard errors are reported over each task; that is, with 5 data sets and 20 tasks per data set, the standard errors represent the sample standard deviation of 100 values divided by $\sqrt{100}$. *Note:* AndoZhang's NA values are due to the fact that it does not explicitly select features.

Method	Test Error $\mu \pm \sigma$	Coefficient Precision/Recall	Feature Precision/Recall
Partial Synthetic Dataset			
True Model	0.07 ± 0.00	$1.00 \pm 0.00/1.00 \pm 0.00$	$1.00 \pm 0.00/1.00 \pm 0.00$
Partial MIC-Multi	$\mathbf{0.10 \pm 0.00}$	$0.84 \pm 0.02/0.77 \pm 0.02$	$0.99 \pm 0.01/0.54 \pm 0.05$
Full MIC-Multi	0.17 ± 0.01	$0.26 \pm 0.01/0.71 \pm 0.03$	$0.97 \pm 0.02/0.32 \pm 0.03$
Baseline (Independent)	0.12 ± 0.01	$0.84 \pm 0.02/0.56 \pm 0.02$	$0.72 \pm 0.05/0.62 \pm 0.04$
BBLasso	0.19 ± 0.01	$0.04 \pm 0.00/0.81 \pm 0.02$	$0.20 \pm 0.03/0.54 \pm 0.01$
AndoZhang	0.50 ± 0.02	NA	NA
Full Synthetic Dataset			
True Model	0.07 ± 0.00	$1.00 \pm 0.00/1.00 \pm 0.00$	$1.00 \pm 0.00/1.00 \pm 0.00$
Partial MIC-Multi	$\mathbf{0.08 \pm 0.00}$	$0.98 \pm 0.01/1.00 \pm 0.00$	$0.80 \pm 0.00/1.00 \pm 0.00$
Full MIC-Multi	$\mathbf{0.08 \pm 0.00}$	$0.80 \pm 0.00/1.00 \pm 0.00$	$0.80 \pm 0.00/1.00 \pm 0.00$
Baseline (Independent)	0.11 ± 0.01	$0.86 \pm 0.02/0.63 \pm 0.02$	$0.36 \pm 0.06/1.00 \pm 0.00$
BBLasso	0.09 ± 0.00	$0.33 \pm 0.03/1.00 \pm 0.00$	$0.33 \pm 0.17/1.00 \pm 0.00$
AndoZhang	0.45 ± 0.02	NA	NA
Independent Synthetic Dataset			
True Model	0.07 ± 0.00	$1.00 \pm 0.00/1.00 \pm 0.00$	$1.00 \pm 0.00/1.00 \pm 0.00$
Partial MIC-Multi	0.17 ± 0.01	$0.95 \pm 0.01/0.44 \pm 0.02$	$1.00 \pm 0.00/0.44 \pm 0.02$
Full MIC-Multi	0.36 ± 0.01	$0.06 \pm 0.01/0.15 \pm 0.02$	$1.00 \pm 0.00/0.14 \pm 0.02$
Baseline (Independent)	$\mathbf{0.13 \pm 0.01}$	$0.84 \pm 0.02/0.58 \pm 0.02$	$0.83 \pm 0.02/0.58 \pm 0.03$
BBLasso	0.35 ± 0.01	$0.02 \pm 0.00/0.43 \pm 0.02$	$0.30 \pm 0.05/0.42 \pm 0.06$
AndoZhang	0.49 ± 0.00	NA	NA

Yeast Dataset Our Yeast dataset comes from (Perlstein et al., 2007). 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).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 $p = 6715$ features.

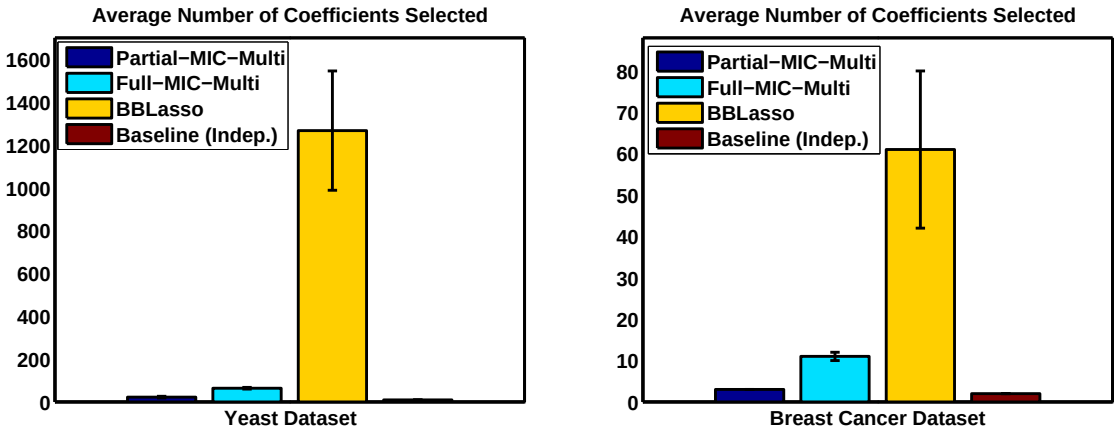
Figure 1 (a) shows classification test errors from 5-fold CV on this data set. As can be seen from the table, Partial MIC-Multi performs better than BBLasso or AndoZhang. Baseline and Full MIC-Multi perform slightly worse than Partial MIC-Multi, under-

scoring the point that it is preferable to use a more general MIC coding scheme compared to Full MIC-Multi or Baseline. The latter methods have strong underlying assumptions, which cannot always correctly capture sharing across tasks.



(a) (b)

Figure 1: Accuracy and number of features selected on five folds of CV for the Yeast and Breast Cancer data sets. *Note:* 1). AndoZhang's average number of features selected are not present in the graph as it does not explicitly select features. 2). These are true cross-validation accuracies and no parameters have been tuned on them.



(a) (b)

Figure 2: Number of coefficients selected on five folds of CV for the Yeast and Breast Cancer data sets. *Note:* 1). AndoZhang's average number of coefficients selected are not present in the graph as it does not explicitly select features.

Breast Cancer Dataset Our second data set pertains to Breast Cancer, and contains data from five of the seven data sets used in (van 't Veer, 2002). It contains 1171 obser-

vations 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 $p = 5000$ (the 5000 features with the highest variance), to save computational resources. Figure 1 (a) shows classification test errors from 5-fold CV on this data set. As is clear from the table, Partial MIC-Multi 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.

The number of features and coefficients selected by all the methods are shown in Figures. 1 (b) and 2 respectively.

5.2 MIC-Single

In this section we demonstrate the results of the MIC-Single scheme on synthetic and real world datasets. For our experiments we use both the MIC-Single (I) and MIC-Single (II) (as described in Algorithm 2) methods and compare against Baseline Feature Selection (which in this case is equivalent to a RIC penalized regression and has a coding scheme similar to Equation 10, Lasso (Tibshirani, 1996), Elastic Nets (Zou and Hastie, 2005) and Group Lasso/ Multiple Kernel Learning (Yuan and Lin, 2006; Jacob et al., 2009; Bach et al., 2004).

For Group Lasso/Multiple Kernel Learning[13], we used a set of 13 candidate kernels, consisting of 10 Gaussian Kernels (with bandwidths $\sigma = 0.5 - 20$) and 3 polynomial kernels (with degree 1-3) for each feature class as is done by (Rakotomamonjy et al., 2008). In the end the kernels which have non zero weights are the ones that correspond to the selected feature classes. Since GL/MKL minimizes a mixed $\ell_1 - \ell_2$ norm so, it zeros out some groups (feature classes)[14]. (Recall that GL/MKL gives no sparsity at the level of features within a feature class). The Group Lasso(Yuan and Lin, 2006; Jacob et al., 2009) and Multiple Kernel Learning are equivalent, as has been mentioned in (Bach, 2008), therefore we used the *SimpleMKL* toolbox (Rakotomamonjy et al., 2008) implementation for our experiments. For Lasso and Elastic Nets we used their standard LARS (Least Angle Regression) implementations (Efron et al., 2004). When running Lasso and Elastic Nets, we pre-screened the datasets and kept only the best $\sim$ 1,000 features (based on their p-values), as otherwise LARS is prohibitively slow. (The authors of the code we used do similar screening, for similar reasons.) For all our experiments on Elastic Nets (Zou and Hastie, 2005) we chose the value of λ_2 (the weight on the ℓ_2 penalty term), as 10^{-6}. We also compared our results with a thresholded version of Group Lasso in which all the weights below a certain cross-validated threshold were set to zero (Zhou, 2009). Although the number of features selected reduced significantly by using this procedure as now we have sparsity within groups also, but the

13. There is a similar relation between MIC-Single and GL/MKL as it is between MIC-Multi and BBLasso. Both BBLasso and GL/MKL are ℓ_1/ℓ_2 penalty based methods and try to solve the same sparsity problem as the corresponding MIC method.
14. However it is possible to estimate the exact support by thresholding (cross-validated) the estimated weights, as has been done by (Zhou, 2009; Lounici, 2008), and enforce sparsity within the groups also.

predictive accuracy did not change significantly so we do not report test accuracies for this thresholded procedure but we do report the number of features selected.

We demonstrate the effectiveness of MIC-Single on synthetic datasets and on real datasets pertaining to Word Sense Disambiguation (WSD) (Chen and Palmer, 2005) (OntoNotes Dataset (Hovy et al., 2006)) and gene expression data (Mootha, 2003).

5.2.1 Evaluation on Synthetic Datasets

The main hypothesis is that MIC-Single methods are beneficial when some groups have multiple predictive features, while others lack them. MIC-Single is particularly effective when there are small groups which contain highly predictive features and big groups containing no predictive features.

In order to validate our hypothesis, we test MIC-Single on two synthetic datasets. For both the datasets, 1000 features were generated independently from a Normal Distribution $\mathcal{N}(0,1)$, and the response vector of 100 observations Y was computed as the linear combination of a set of 7 beneficial features and Gaussian additive noise ($\mathcal{N}(0,1.7^2)$). The first data set (Set 1) had 4 groups (feature classes) of unequal sizes and 7 beneficial features, all of which lie in a small feature class of size 12. The second synthetic dataset (Set 2) was generated so as to reflect the other extreme case, in which all the classes are of same size, and had 100 feature classes, each of size 100. Again all 7 beneficial features were in a single feature class.

Table 3: The number of correct and spurious Features Selected and Ten Fold Cross Validation (CV) Test Errors averaged over 10 runs. Set 1). Unequal class sizes, Set 2). Uniform class sizes. **Note:** We did not compare against Group Lasso/MKL on synthetic datasets as we feel that it will be an unfair comparison favoring MIC methods as even the thresholded versions of GL/MKL added lots of features and were not competent with other methods in terms of sparsity induced.

Method	Avg. Features Selected				10-Fold CV Error	
	Correct		Spurious			
	Set 1	Set 2	Set 1	Set 2	Set 1	Set 2
MIC-Single (II)	**6.8 ± 0.1**	**5.6 ± 0.0**	**0.1 ± 0.0**	0.3 ± 0.1	**0.09 ± 0.02**	**0.27 ± 0.01**
MIC-Single (I)	6.7 ± 0.0	5.4 ± 0.1	**0.1 ± 0.1**	0.2± 0.1	0.11 ± 0.02	0.28 ± 0.02
Lasso	5.2 ± 1.0	4.3 ± 1.2	2.2 ± 1.0	1.8 ± 0.1	0.22 ± 0.03	0.41 ± 0.02
Elastic Nets	6.4 ± 0.2	4.9 ± 0.7	3.3± 1.1	2.1 ± 1.3	0.20 ± 0.03	0.43 ± 0.02
Baseline (RIC)	4.4 ± 1.4	3.2 ± 2.2	0.2 ± 0.1	**0.0 ± 0.0**	0.27 ± 0.05	0.61 ± 0.04

As can be seen from the results in Table 3, in both cases the MIC-Single methods outperform other competing methods.

5.2.2 Evaluation on Real Datasets

In order to benchmark the real world performance of our MIC-Single, we chose two datasets pertaining to two diverse applications of feature selection methods, namely Computational

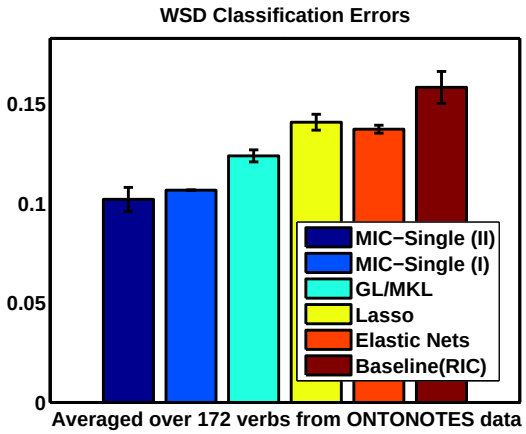


Figure 3: 10 Fold CV classification test accuracies averaged over 172 verbs. *Note:* 1). These are true cross-validation accuracies and no parameters have been tuned on them.

Linguistics and Gene Expression Analysis. More information regarding the data and the experimental results are given below.

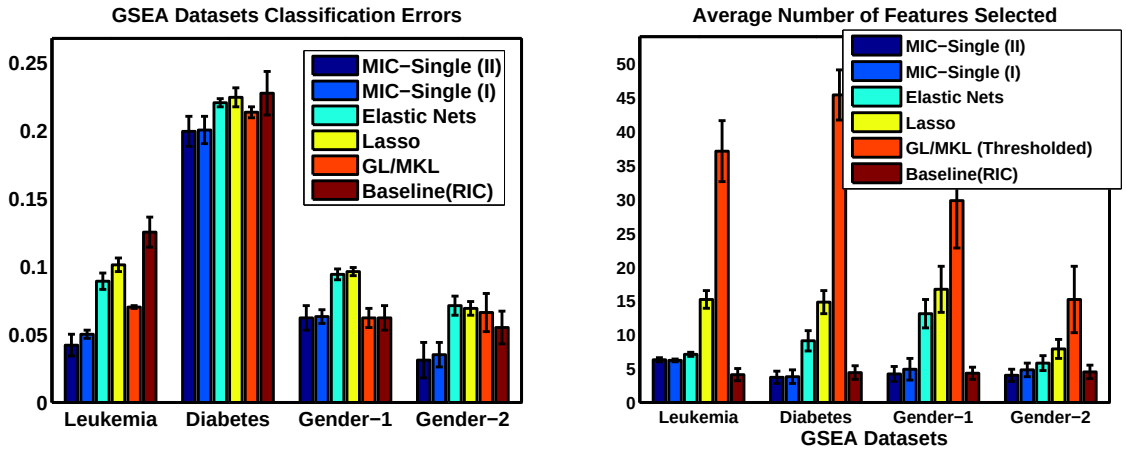
Word Sense Disambiguation (WSD) Datasets: A WSD dataset (OntoNotes (Hovy et al., 2006)) consisting of 172 ambiguous verbs and a rich set of contextual features (Chen and Palmer, 2005) was chosen for evaluation. It consists of hundreds of observations of noun-noun collocation, noun-adjective-preposition-verb (syntactic relations in a sentence) and noun-noun combinations (in a sentence or document).

The dataset had a total of 172 verbs with $40 - 45$ feature classes (groups). The number of observations n for the various verbs varied from 100 to 3500 and the number of features p varied from 1000 to 11500.

The classification[15] test accuracies averaged over all the 172 verbs[16] are shown in Figure 3. Thresholded version of Group Lasso also gave similar predictive accuracy as the actual Group Lasso but it gave much more sparser models. It is also worth noting that MIC-Single (II) was ~ 7 times faster than MIC-Single (I) as we had hypothesized earlier, as for each selected feature it does a subset search within that feature's group (feature class) to find the optimal number of features to select from the group.

Gene Set Enrichment Analysis (GSEA) Datasets: The second real datasets that we used for our experiments were gene expression datasets from GSEA (Mootha, 2003). There are multiple gene expression datasets and multiple criteria on which the genes can be grouped into classes. For example, different ways of generated gene classes include C1: Positional Gene Sets, C2: Curated Gene Sets, C3: Motif Gene Sets, C4: Computational Gene Sets, C5: GO Gene Sets.

15. As with MIC-Multi we used MIC-Single to do feature selection and once we had selected the features we used logistic regression for the final classification problem.
16. These accuracies are for the (one vs all) binary class prediction problem i.e. predicting the most frequent sense. On the entire set of 172 verbs, MIC-Single methods are significantly (5 % significance level (Paired t-Test)) better than the competing methods on 160/172 verbs and have the same accuracy as the best method on 4 occasions.



(a) (b)

Figure 4: 10-fold CV classification test accuracies and the average number of features selected by various methods on the GSEA datasets. *Note:* 1). These are true cross-validation accuracies and no parameters have been tuned on them.

For our experiments, we used gene classes from the C1 and C2 collections. The gene sets in collection C1 consists of genes belonging to the entire human chromosome, divided into each cytogenetic band that has at least one gene. Collection C2 contained gene sets from various sources such as online pathway databases and knowledge of domain experts.

The datasets that we used and their specifications are as shown in Table 4.

Table 4: GSEA Datasets.

Dataset	# Observations (n)	# Features (p)	# Classes (K)
Leukemia (C1)	48	10056	182
Gender 1 (C1)	32	15056	212
Diabetes (C2)	34	15056	318
Gender 2 (C2)	32	15056	318

The results for these GSEA datasets are as shown in the Figure 4.

For these datasets also MIC-Single methods also beat the competing methods. Here also MIC-Single is significantly (5% significance level, Paired t-test) better than the competing methods. It is interesting to note that MIC-Single methods sometimes selected substantially fewer features, but still gave better performance than other methods which goes onto show that adding all or many features from a single group contributes to a redundant signal and efficient feature selection "within" a group (feature class) is warranted.

6. MIC Model Consistency

In this section we show that our MIC methods based on two part MDL and with the model coding costs as described in Section 4 are consistent. By "consistent" we mean that if the data is distributed by one of the probabilistic sources in the set of candidate model classes

that our MDL based estimators consider ($\mathcal{M}$), then given enough data, MIC will output the true distribution generating the data. The proof of consistency is similar to the proof of classical two part MDL consistency as given in (Barron and Cover, 1991) and the recent improvement to that proof by (Zhang, 2004) by using ideas from KL-complexity[17]. To extend these proofs to the case of MIC, first we will explain the concepts of *Universal Codes* and *Asymptotic Distinguishability* and also state the *no-hypercompression inequality* from information theory.

We first define some common notation that will be useful throughout this section. Assume we have n data samples (observations) $\{X_1, \ldots, X_n\} \in \mathcal{X}$ and that they are distributed according to some distribution P_{True}. Further let P_{Model} be an arbitrary distribution on $\mathcal{X}$ (the distribution estimated by our MIC based model). Also, as shorthand we denote $-\sum_{i=1}^{n} \lg Q(X_i)$ as $-\lg Q(X^n)$ throughout this section.

No Hypercompression Inequality:

$$\forall K > 0, P_{True}\left[-\lg(P_{True}(X^n)) \geq -\lg(P_{Model}(X^n)) + K\right] \leq 2^{-K} \tag{26}$$

This inequality states that the probability of a code not corresponding to P_{True} compressing the data by more than K bits than the code corresponding to P_{True} does is exponentially small in K, where K is any positive number.

The proof follows by using Markov's inequality and can be found in (Grünwald, 2007).

Asymptotic Distinguishability: If the actual data was generated by the distribution P_{True} then the distribution P_{Model} is said to be *asymptotically distinguishable* from P_{True} if the probability that a sample generated by P_{True} looks as if it were generated by P_{Model} rather than P_{True} approaches 0 with increasing sample size. More formally,

$$\forall \delta > 0, P_{True}\left[\frac{P_{Model}(X^n)}{P_{True}(X^n)} > \delta\right] \rightarrow 0 \; as \; n \rightarrow \infty \tag{27}$$

In other words, if P_{Model} is not asymptotically distinguishable from P_{True} then even if we observe an infinite amount of data, we cannot distinguish whether the data was drawn from P_{True} or from P_{Model}. So, we should be satisfied if given enough data, any learning algorithm outputs as its hypothesis any distribution (model) that is asymptotically indistinguishable from P_{True}.

Universal Coding Schemes: Going back to the standard MDL setting which envisions a Sender and Receiver, assume that the Sender and Receiver have a set of candidate coding schemes $\mathcal{L}$ for $\mathcal{X}^n$ available. Both of them know that one of these available codes will give the highest compression for the sequence $X^n \in \mathcal{X}^n$. In other words:

$$L_{Optimal}(X^n) = \underset{i}{\operatorname{argmin}}\{L_i(X^n)\} \; \forall L_i \in \mathcal{L} \tag{28}$$

However, they must decide on a code before the sender observes the actual data X^n and they do not know which is the best code[18]. One thing that the Sender can do is on seeing

17. (Zhang, 2004) also outlined the conditions under which Hellinger Consistency can not be obtained for two part MDL but "weak consistency" is possible.

18. In the Bayesian terminology, this problem is similar to finding the classifier which has the Optimal Bayes Risk–i.e., the classifier with the minimum possible risk among all the candidate classifiers.

the data (X^n), he encodes the data using $L_{Optimal}$ as described above. However, this is not feasible as the Receiver does not know what code the Sender used and so he would not be able to decode the message. Therefore it is not possible to find the best code that compresses the data and so in practice people use *universal codes* which compress the data almost as well as $L_{Optimal}$[19]. It has been shown that the two part MDL codes that we used in this paper to describe MIC based methods are universal codes (Grünwald, 2005; Grünwald, 2007; Rissanen, 1999). Moreover the "uniform prior" code[20] and the "combinatorial code" we used in coding the model for MIC-Multi and MIC-Single are also universal codes (Grünwald, 2007).

Consistency Results: As mentioned earlier, two part MDL has been proved to be consistent in a variety of settings (Barron and Cover, 1991; Zhang, 2004; Grünwald, 2007). Here we provide similar proofs for the case of our MIC based models.

Let $\mathcal{M}$ be a countable number of possible distributions that our data ($\mathcal{X}$) can belong to and let $P_{True} \in \mathcal{M}$[21] and let ℓ be some codelength function, corresponding to a code over elements of $\mathcal{M}$. As earlier, the data is distributed according to P_{True}. Also, let P_n^{MIC} be the distribution corresponding to the two part MDL model selected by MIC.

Theorem 1 *Let $\mathcal{M}' \subset \mathcal{M}$ be the set of sources in $\mathcal{M}$ that are asymptotically distinguishable from P_{True} and $\ell(P_{True}) < \infty$, then*

$$P_{True}(P_n^{MIC} \in \mathcal{M}') \to 0 \; as \; n \to \infty \tag{29}$$

The theorem states that the probability that MIC selects a probabilistic source to explain the data that is asymptotically distinguishable from the true underlying distribution (P_{True}) approaches 0 as the number of observations increase. In other words, with overwhelming probability, P_{True} is asymptotically indistinguishable from P_n^{MIC} as n approaches infinity.
Proof

$$\begin{aligned} &P_{True}\left[P_n^{MIC} \in \mathcal{M}'\right] = \\ &\quad P_{True}\left[\; for \; some \; P_n^{MIC} \in \mathcal{M}' : \ell(P_{True}) + \ell(X^n|P_{True}) \geq \ell(P_n^{MIC}) + \ell(X^n|P_n^{MIC})\right] \\ &\quad \leq \sum_{P_n^{MIC} \in \mathcal{M}'} \left[\ell(P_{True}) + \ell(X^n|P_{True}) \geq \ell(P_n^{MIC}) + \ell(X^n|P_n^{MIC})\right] \end{aligned} \tag{30}$$

The above inequality is obtained by applying the Union Bound over all the possible models that P_n^{MIC} can output from $\mathcal{M}'$.

Now, by re-arranging Equation 30 and noting that the error term in two-part coding can be replaced by a log term as in Equation 4,

$$g_n(P_n^{MIC}) = P_{True}\left[-\lg(P_{True}(X^n)) \geq -\lg(P_n^{MIC}(X^n)) + \ell(P_n^{MIC}) - \ell(P_{True})\right] \tag{31}$$

19. This corresponds to a classifier whose risk is close to Bayes Risk.
20. As mentioned earlier, the uniform code is also minimax optimal.
21. We assume that the true underlying distribution belongs to the set of distributions that our models consider. (Zhang, 2004; Barron and Cover, 1991) also make this assumption.

In the case of MIC based methods, the $\ell(P_n^{MIC})$ term corresponds to the number of bits required to code the models for MIC-Multi and MIC-Single (II) as mentioned in Equations (23),(15) and similarly the $-\lg(P_n^{MIC}(X^n))$ term corresponds to the likelihood terms in the MIC models.

Applying the Equation 30 to Equation 31, we can write:

$$P_{True}\left[P_n^{MIC} \in \mathcal{M}'\right] \leq \sum_{P_n^{MIC} \in \mathcal{M}'} g_n(P_n^{MIC})$$

Using the no-hypercompression inequality with $K = \ell(P_n^{MIC}) - \ell(P_{True})$, we get

$$g_n(P_{Model}) = P_{True}\left[-\lg(P_{True}(X^n)) \geq -\lg(P_n^{MIC}(X^n)) + \ell(P_n^{MIC})\right] \leq 2^{-\ell(P_n^{MIC})+\ell(P_{True})}$$

Finally, by Kraft's inequality (Grünwald, 2005) we know that for any legitimate coding scheme ($\ell(\cdot)$) we have $\sum_{x \in \mathcal{M}} 2^{-\ell(x)} \leq 1$

$$\sum_{P_n^{MIC} \in \mathcal{M}'} g_n(P_n^{MIC}) \leq \sum_{P_n^{MIC} \in \mathcal{M}'} 2^{-\ell(P_n^{MIC})+\ell(P_{True})} \leq \theta \cdot 2^{\ell(P_{True})},$$

where $\theta \leq 1$. Now, if we define $\theta = \epsilon \cdot 2^{-\ell(P_{True})}$, we can see that

$$\lim_{n \to \infty} P_{True}\left[P_n^{MIC} \in \mathcal{M}'\right] < \epsilon$$

which proves the theorem.

■

A corollary of the above theorem is that the MIC coding schemes as described in Section 4 are not arbitrary. If we design really bad coding schemes we may need large numbers of data observations for MIC to converge to a distribution that is asymptotically indistinguishable from P_{True}. However, since our MIC based coding schemes use *universal codes*, which are shown to be very close approximations to the shortest possible code $L_{Optimal}$ as described above, we do not need a very big sample size for consistency. This underscores the point that we cannot tweak MDL by using arbitrary codes to give the answers that we would like it to give.

Another important theoretical property that is attractive for sparse learning algorithms is *sparsistency*, which is shorthand for "sparsity pattern consistency". In other words:

$$P\left[supp(w^{True}) = supp(w_n^{MIC})\right] \to 1 \; as \; n \to \infty,$$

where $supp(w) = \{w : w_j \neq 0\}$, w^{True} is the true sparse weight vector and w^{MIC} is the weight vector estimated by MIC based methods. Sparsistency implies that the learning algorithm is consistently able to identify the correct set of sparse features in the asymptotic limit.

Lasso and Group Lasso have been proved to be *sparsistent* under irrepresentable conditions that depend on the sign of the true weight vector (w^{True}) (Zhao and Yu, 2006; Wainwright, 2009; Meinshausen and Bühlmann, 2006; Bach, 2008). (Tropp, 2004) proved

that forward greedy feature selection also selects features consistently when the linear model has a zero-mean stochastic noise; (Zhang, 2009b) improved this result to include non-zero mean sub-Gaussian stochastic noise. However, due to the complexity of the forward greedy feature selection the sparsistency condition in this case depends only on the feature (design) matrix X, unlike Lasso and Group Lasso.

Since our MIC based methods are based on forward greedy feature selection, i.e., they use the MDL principle to provide a cost function which is greedily minimized by a forward search, they should be *sparsistent*. However, for ℓ_0 penalized regression, the *sparsistency* condition also depends on the information theoretic penalty in that the penalty must increase with n (the number of observations) (Wu and Zhou, 2010). For our MIC based methods this penalty is a combination of RIC, AIC (to code the coefficients) and other coding schemes which incorporate the structure of the problem at hand. The penalties for the MIC based methods as presented in this paper do not have the required dependence on n, so they are not *sparsistent*. However, we could modify our coding schemes slightly by using the BIC penalty ($\lg(n)$ bits) to code the coefficients instead of AIC to ensure *sparsistency* of MIC. However, we prefer that our methods are not *sparsistent* as in that case we achieve competitive performance with the true underlying model, i.e., we get finite *risk-inflation* of about $2\lg(p)$ (Foster and George, 1994) whereas if we chose *sparsistency* then MIC would have infinite *risk-inflation*. Thus, given the choice between *better model-fit* and *sparsistency*, we chose the former. However, if *sparsistency* is more important than predictive accuracy, making a small change in the coding schemes would guarantee it.

7. A Model for "Intra Domain" adaptation: TransFeat

In the previous sections we proposed MIC based methods for the related problems of simultaneous feature selection for a set of multiple related tasks (MIC-Multi) and grouped feature selection for single task (MIC-Single). The focus of those methods was joint feature selection, but in many applications it is the case that some of the tasks have less data available than other tasks and building supervised learning models from the limited amount of data does not give high predictive accuracies. So, it becomes desirable to "borrow strength" for the tasks with less amount of data from the tasks with lots of data. In other words, we want to have "intra domain" adaptation or Transfer Learning (Ando and Zhang, 2005; Raina et al., 2006).

In this section, we propose a method called TransFeat which addresses the above problem by transferring information between similar tasks by using a feature relevance prior. We demonstrate the effectiveness of TransFeat for the problem of Word Sense Disambiguation (WSD), and show that in this domain TransFeat significantly improves accuracy on tasks with less data. TransFeat, could, of course, be applied to wide variety of domains, but is particularly useful for WSD as state-of-the-art WSD systems, including the ones that use feature selection, are strongly limited by the paucity of labeled data. For example, the training set of the Senseval-2 English lexical sample task has only ~ 10 labeled examples per sense (Florian and Yarowsky, 2002). Such limited data makes it difficult to build high accuracy models using standard supervised learning techniques and suggests the use of transfer learning to improve performance.

As mentioned above, TransFeat learns a feature relevance prior from "similar" tasks, and gives supervised learning accuracies which are comparable to or better than state-of-the-art WSD systems. Learning this prior for feature relevance of a test task makes those features that have been selected in the models of other "similar" tasks become more likely to be selected. TransFeat does this by using a MDL-based approach similar to the MIC methods presented above.

Task Setting: We are given a set of target words each having an $n \times p$ feature matrix ($\mathbf{X}_{n\times p}$), where n is the total number of observations (instances) and p is the total number of features. We have a $n \times h$ response matrix ($\mathbf{Y}_{n\times h}$) of the h sense labels for each of the n observations. The WSD task is to assign a sense to each test instance[22].

Overview of TransFeat: TransFeat builds upon MIC-Single and it has several steps:

- Break the $Y_{n\times h}$ matrix into h, $n \times 1$ matrices i.e., out of one multiclass (h classes) problem we make h binary class problems. The prediction problem now becomes "Is this word sense 1 or not?", etc. The main reason for doing this is that not all senses of all words are similar to all senses of some other word. Thus, transfer learning only makes sense at level of individual word *senses* rather than at the level of whole words.

- Make separate feature matrices for these h prediction problems, because the original feature matrix $X_{n\times p}$ contained features which would be useful for the multiclass problem of "What is the exact sense of the word?", rather than for the binary problems of "Is this sense 1 or not?" and so on. We do this is by characterizing each binary problem by those features from the original p features which are positively correlated with that particular word sense[23]. This gives h feature matrices $X_{\{i=1,\ldots,h\}}$ drawn from the original $n \times p$ feature matrix, where each of these matrices need not have the same number of features.

- Next, cluster the different word senses by using "forward-background" clustering that puts all singleton points into a "background cluster" which we then ignore[24].

- Learn a separate MIC-Single (II) model for each word sense[25].

- For each word sense in a cluster, use TransFeat to learn a feature relevance prior from the remaining word senses in that cluster that have more observations than the target word sense, on the features of that word sense. As we explain later, this feature relevance prior allows us to learn better MIC-Single (II) models by relaxing the *uniform prior* assumption that each group (feature class) and then each feature within that group is equally likely to be selected, that MIC-Single (II) makes.

22. Note that this is a multi-class problem; we have a single task, which is to predict the correct sense of the word and we have h possible choices (the word senses) for that task. So, we approach it differently from the multi-task problem (MIC-Multi), where we predicted all tasks jointly.
23. In general, features with positive coefficients are associated with the given sense and those with negative coefficients with other senses of that word.
24. Later we explain in detail why standard clustering methods like k-means fail in this case.
25. As mentioned in the Section about MIC-Single, WSD is one problem which exhibits group structure and therefore we use it as a base model on which we build TransFeat.

- Given these better MIC-Single models for all the word senses, we solve the actual h class WSD problem by choosing the sense whose model gave the highest score as the most likely sense for that word.

We learn the feature relevance prior only from distributionally similar word *senses*; in contrast to (Ando, 2006) who share knowledge across "all" the senses of "all" the words. Our approach makes sense as it is difficult to find words which are similar in all their senses. We can, however, often find words which have one or a few similar senses. For example, one sense of "fire" (as in "fire someone") should share features with one sense of "dismiss" (as in "dismiss someone"), but other senses of "fire" (as in "fire the gun") do not. Similarly, other meanings of "dismiss" (as in "dismiss an idea") should not share features with "fire". Similarly, the words "kill", "capture" and "arrest," share one similar sense. This justifies our choice of breaking down the problem down to the level of individual word senses.

Thus, knowledge can only be fruitfully transferred between the shared senses of different words, even though the models being learned are for disambiguating different senses of a single word. To address this problem, we cluster similar word senses of different words, and then use the models learned for all the word senses in the cluster with more data (observations) than the held out word sense (called "training word senses") to put a feature relevance prior on what features will be more predictive for the held out test word sense. We hold out each word sense in the cluster once and learn a prior from the remaining word senses in that cluster. For example, we can use the models for discriminating the senses of the words "kill" and the senses of "capture", to put a prior on what features should be included in a model to disambiguate senses of the distributionally similar word "arrest", which has considerably less data than the other two words (OntoNotes dataset), hence enabling us to learn high accuracy models for "arrest". If at least one sense of the word "arrest", that we are trying to model is similar to the other word senses (for "kill" and "capture"), some of the same features should be beneficial for all of them.

7.1 TransFeat Formulation

We now describe TransFeat in detail and show how it can be used to learn better feature selection models by relaxing the overly simplistic assumption of the model coding schemes of MIC methods of *uniform prior* by learning a feature relevance prior.

We define a binary random variable $f_i \in \{1,0\}$ that denotes the event of the i^{th} feature being in or not being in the model for the test word sense, and model it as being from a Bernoulli distribution parametrized by θ_i:

$$p(f_i|\theta_i) = \theta_i^{f_i}(1-\theta_i)^{1-f_i} \tag{32}$$

Given the data for the i^{th} feature for all the training word senses, we can write: $\mathcal{D}_{f_i} = \{f_{i1}, ..., f_{iv}, ..., f_{it}\}$. We then construct the likelihood functions from the data (under the i.i.d assumption) as:

$$p(\mathcal{D}_{f_i}|\theta_i) = \prod_{v=1}^{t} p(f_{iv}|\theta_i) = \prod_{v=1}^{t} \theta^{f_{iv}}(1-\theta_i)^{1-f_{iv}}$$

The posteriors can be calculated by putting a prior over the parameters θ_i and using Bayes rule as follows:

$$p(\theta_i|\mathcal{D}_{f_i}) = p(\mathcal{D}_{f_i}|\theta_i) \times p(\theta_i|a,b) \tag{33}$$

where a and b are the hyperparameters of the Beta Prior (the conjugate of the Bernoulli distribution). The predictive distribution of θ_i is:

$$p(f_i = 1|\mathcal{D}_{f_i}) = \int_0^1 p(f_i = 1|\theta_i)p(\theta_i|\mathcal{D}_{f_i})d\theta_i \tag{34}$$

Substituting from 32 in the above equation we get:

$$p(f_i = 1|\mathcal{D}_{f_i}) = \int_0^1 \theta_i p(\theta_i|\mathcal{D}_{f_i})d\theta_i = \mathbb{E}[\theta_i|\mathcal{D}_{f_i}] \tag{35}$$

Using the standard results for the mean and the posterior of a Beta distribution we obtain:

$$p(f_i = 1|\mathcal{D}_{f_i}) = \frac{k+a}{k+l+a+b} \tag{36}$$

where k is the number of times that the i^{th} feature is selected and l is the complement of k, i.e. the number of times the i^{th} feature is not selected in the training data.

As can be seen from Equation 36, the probability that a feature is selected for the held out test word sense is a "smoothed" average of the number of times it was selected in the models for the senses of other words that are similar to it.

Using similar reasoning, we can extend the above concept to the groups (feature classes) so that the probability that a group (feature class) is selected is also a "smoothed" average of the number of times it was selected in the models for the senses of other words that are similar to it.

In light of the above reasoning, the modified model cost for MIC-Single for coding the i^{th} feature when previously no features have been selected from the j^{th} feature class which contains that feature can be written as follows:

$$S_M^i = -\lg\left(p(G_j = 1|\mathcal{D}_{G_j})\right) - \lg\left(p(f_i = 1|\mathcal{D}_{f_i})\right) + 2$$

and for the case when some features have already been selected from the j^{th} feature class, we can write a modified coding cost as follows:

$$S_M^i = \min\left[-\lg\left(p(G_j = 1|\mathcal{D}_{G_j})\right), 1 + \lg(Q)\right] - \lg\left(p(f_i = 1|\mathcal{D}_{f_i})\right) + 2$$

where the first term represents the probability of selecting at least one feature from the j^{th} feature class, the second term represents the probability of selecting the i^{th} feature, and the third term which is used to code the coefficient values remains the same as earlier[26]. Note that in the case when we have previously selected features from a given feature class, the most efficient way to code the feature class is to use the minimum of the TransFeat cost and the actual "switch" coding cost as described in Section 4.2. Thus TransFeat replaces the implicit uniform prior of MIC-Single with a coding scheme which is more informed by the prior learned from similar tasks.

The detailed algorithm for TransFeat is given below in Algorithm 3.

26. The negative sign is due to the duality between Bayesian and Information Theoretic interpretation as mentioned earlier.

Algorithm 3 TransFeat Algorithm

Break the multiclass problem into h binary prediction problems.
Make the feature matrices for each of these problems i.e. $X_{\{i=1,\ldots,h\}}$.
Cluster the different word senses by "foreground-background" clustering.
$total\_clusters = \{1, \ldots, c\}$
$word\_senses_k = s_k$ // Number of word senses in k^{th} cluster.
for i in $total\_clusters$ **do**
  for t in $word\_senses_i$ **do**
    Learn MIC-Single (II) model for all the word senses. // Uniform prior assumption
  end for
end for
for i in $total\_clusters$ **do**
  for t in $word\_senses_i$ **do**
    Learn TransFeat model on all word senses in the cluster which have more data (observations) than the t^{th} word sense.
    Use the revised model costs S_M output by TransFeat to learn better MIC-Single (II) model for t^{th} word sense.
    // The uniform prior assumption of MIC-Single (II) has been relaxed.
  end for
end for
Disambiguate the word as a whole by choosing the correct sense (from h possible senses) as the one whose model gave the highest score.

7.1.1 Choice of Hyperparameters

The hyperparameters a and b in Equation 36 control the "smoothing" of our probability estimates, i.e., how strongly we want the evidence obtained from similar word senses to effect the model that we learn for the test word sense.

In all our experiments we set $a = 1$ and choose b so that in the limiting case of no transfer i.e. ($k = l = 0$ in Equation 36) the coding scheme will reduce to the baseline feature selection described in (Equation 10). Thus, we choose $b = p - 1$ where p is the total number of features/feature classes (depending on what we are coding) in the test word sense.

7.2 Experimental Results

In this section we first describe our data and similarity metric that we used; we then report the results of applying TransFeat to the Senseval-2 and OntoNotes datasets.

7.2.1 Similarity Metric

Finding a good similarity metric between different word senses is perhaps one of the biggest challenges that we faced. It is also the part of this section that is specific to the problem of word sense disambiguation. There are many ways in which word senses can be judged as similar, including having similar "meanings" or similar syntactic usages. Human annotated lexicons such as Levin classes (Levin, 1993), hypernyms or synonyms according to WordNet (Miller, 1990; Lin, 1999), or VerbNet classes (Kipper et al., 2000; Schuler, 2006) capture different aspects of this similarity, as does InfoMap(http://infomap.stanford.edu) (Raina et al., 2006), which gives a distributional similarity score for words in the corpus. We choose

instead to define a similarity metric based, as described below on combinations of many different aspects of the lexical and syntactic context of the word.

One might think of doing k-means clustering of the word senses based on their features, but this works poorly, as it assigns all the word senses to some cluster, while in reality, there are in practice many word senses that are not sufficiently similar to any other word sense, either semantically or syntactically and hence many word senses occur in "singleton" clusters. K-means and perhaps surprisingly, hierarchical agglomerative clustering, even after extensive use of different "k" or thresholds, failed to give reasonable clusters.

We thus need a clustering method that gives "tight" clusters of word senses, and does not attempt to cluster those word senses which are not similar to any other word sense in the corpus. We do this using a "foreground-background" clustering algorithm as proposed by (Kandylas et al., 2007). This algorithm gives highly cohesive clusters of word senses (the "foreground") and puts all the remaining word senses in the "background". The parameters that it takes as input are the % of data points to put in "background" (i.e., what would be the singleton clusters) and a similarity threshold which impacts the number of "foreground" clusters. We experimented with putting 20% and 33% data points in background and adjusted the similarity threshold to give us $50 - 100$ "foreground" clusters. The results reported below have 20% background and $50 - 100$ "foreground" clusters.

7.2.2 Description of Data

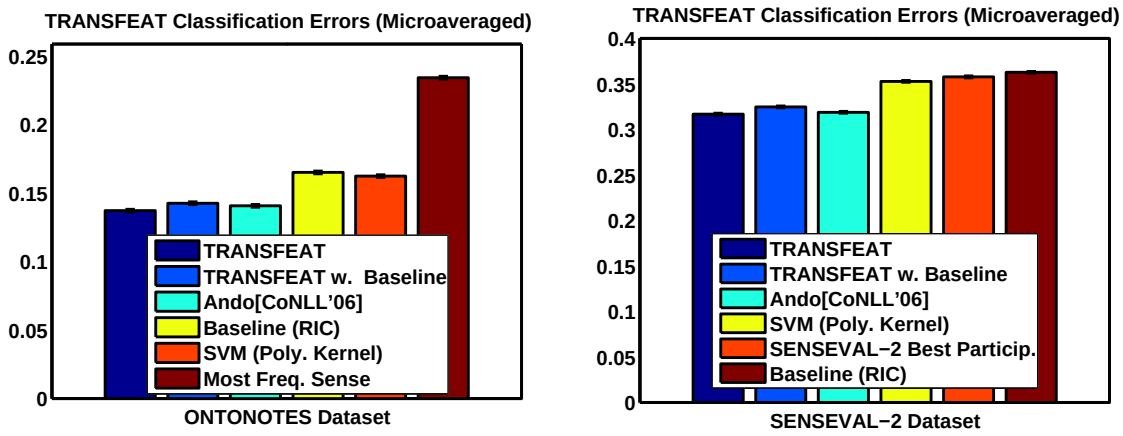
We used the Senseval-2 English lexical sample data, which contains a total of 73 different words (29 nouns, 29 verbs, and 15 adjectives) and the OntoNotes verb data (the same one used for experiments of MIC-Single), containing 172 verbs. The main difference between these two datasets is that Senseval-2 data contains "fine grained" senses of the words and as a result tends to have more senses per word than the "coarse grained" verb senses in OntoNotes. (See Table 5.)

Table 5: Data Statistics of Senseval-2 and OntoNotes data sets. **Note:** In our experiments we used the standard test-train splits for Senseval-2; OntoNotes data does not have any standard splits so we report 10-Fold cross validation test-accuracies

Dataset	#words	#train	avg #senses per word
Senseval-2 (nouns+verbs+adj.)	73	8611	10.7
OntoNotes (only verbs)	172	See Note (*in caption*)	3.7

7.2.3 Results

We cluster the word senses based on all the features, i.e., "semantic+syntactic" similarity features. We experimented clustering using only syntactic and only semantic features but we got the best results using the combined feature set.



(a) (b)

Figure 5: 10-fold CV (micro-averaged) test accuracies of various methods for OntoNotes and Senseval-2 (English Lexical Sample) datasets. **Note:** 1.) These are true cross-validation accuracies and no parameters have been tuned on them. 2.) The final accuracies reported are averaged over the entire 172 verbs. 3). We used the standard test-train splits for SENSEVAL-2 as mentioned on the data website and as used in previous studies

All results reported are micro-averaged[27] accuracies. In order to ensure fairness of comparison we compute the predicted sense for each observation by selecting the word sense model (from among the different senses for that word) with the highest score for that observation sense[28]. This "one vs all" approach to prediction in multi-class problems is widely used, although higher accuracy can sometimes be obtained by more complex round-robin comparison methods.

We use two versions of TransFeat, as can be seen in Figure 5. The first version is exactly the same as mentioned in Algorithm 3, while the second version, TransFeat w. Baseline, builds upon baseline feature selection (Equation 10) instead of MIC-Single (II). We compare TransFeat methods against baseline feature selection (Equation 10), SVM with a polynomial kernel, Ando[CoNLL'06], computed with the standard implementation of the algorithm from the Berkeley Transfer Learning Toolkit (Rakhlin, 2007), and a simple Most Frequent Sense baseline. For SVM we used the standard libSVM package (Chang and Lin, 2001). We used a polynomial kernel, as it gave better performance on held out data than other kernels including linear and RBF. We tuned the cost parameter (c) and the degree of polynomial (d) parameters of the polynomial kernel using a separate cross validation.

27. Our precision and recall are always the same as we assign exactly one sense to each instance. Hence the accuracy that we report is the same as the F-measure or 'micro-averaged' recall as is reported in many WSD studies.
28. As in earlier experiments we use TransFeat only to select features and later we use logistic regression for classification.

7.2.4 Analysis of OntoNotes Results

The results for the different settings for the OntoNotes dataset averaged over the entire 172 verbs are shown in Figure 5 (a). The TransFeat models are significantly better (5% significance level using a paired t-test) than all the competing methods except Ando[CoNLL '06].

Some examples will help to emphasize the point that we made earlier that transfer helps the most in cases in which the target word sense has much less data than the word senses from which knowledge is being transferred. "kill" had roughly 6 times more data than all other word senses in its cluster (i.e., "arrest", "capture", "strengthen", etc.) In this case, TransFeat gave $3.19 - 8.67\%$ higher accuracies than competing methods on these three words. Both versions of TransFeat do much better than Ando[CoNLL'06] on these select words even though on average over all 172 verbs the difference is slender. Similarly, for the case of word "do" which had roughly 10 times more data than the other word senses in its cluster (e.g.,"die" and "save"), TransFeat gave $4.09 - 6.21\%$ higher accuracies than other methods. Transfer makes the biggest difference when the target words have much less data than the word senses they are generalizing from, but even in cases where the words have comparable amounts of data we still get a $1.5 - 2.5\%$ increase in accuracy.

However, as one might expect, transfer learning can sometimes hurt performance; there can be so-called "negative-transfer" (Caruana, 1997). This was the case for 8 verbs out of the 172.

7.2.5 Analysis of Senseval-2 Results

The results for Senseval-2 dataset are shown in Figure 5(b). Here also TransFeat does significantly better (5% significance level using a paired t-test) than the baseline feature selection method and most of the other state-of-the-art algorithms. It is worth noting that a high degree of engineering goes into the state-of-the-art Senseval-2 systems. This is in contrast to TransFeat, which uses information theoretic feature selection and thus has no free parameters to tune. The TransFeat results are comparable to those reported in (Ando, 2006), which is the state-of-the-art system on Senseval-2. Since (Ando, 2006), only mentions the overall accuracy and not the accuracy on individual words, we cannot tell whether this slender difference is statistically significant.

For words that had considerably fewer observations than other words in their cluster, TransFeat again gave major benefits. For example, "begin" had $\sim$ 8 times more data (on average per sense) than the other word senses in its cluster (i.e. "work" and "develop"). In this case, TransFeat gave $6.11 - 7.05\%$ improvement in accuracy over the baseline feature selection. Similarly, "leave" had $\sim$ 2 times more data than "turn" and "strike", and in this case TransFeat gave $5.11 - 6.23\%$ improvement in accuracy over the baseline. These improvements are considerably larger than the average improvement over all the words as reported in Figure 5(b).

For this data set there was negative transfer on 5 out of 73 words.

8. Conclusion

In this paper we presented a framework for learning sparse models based on the information theoretic Minimum Description Length (MDL) principle. We presented two models based on the MIC (Multiple Inclusion Criterion) which greedily select features using the MDL principle in the single and multi task settings respectively. Both the methods, MIC-Multi and MIC-Single, induce a two level sparsity; MIC-Single does feature selection at the level of groups and also at the level of features within each group and MIC-Multi allows each selected feature to be added to the models of some or all of the tasks. We showed how we can use MDL to specify "customized" coding schemes in scenarios where the problem has complex structure. We also discussed the conditions under which the MDL based methods are consistent and *sparsistent* and also showed that the MDL coding schemes are not arbitrary and have a corresponding Bayesian interpretation. Lastly, we proposed a model, TransFeat which can be used to transfer a feature relevance prior to tasks which have less data available. We evaluated all three methods on a variety of domains including genomics (for both yeast and beast cancer) and natural language processing (Word Sense Disambiguation). Our methods are consistently at least as accurate as state-of-the-art methods, while producing models that are more sparse. Such sparseness is particularly important for applications such as genomics and computational linguistics, where interpretable models are valued.

References

H. Akaike. Information theory and the extension of the maximum likelihood principle. In *2nd International Symposium on Information Theory, Budapest*, pages 261–281, 1973.

R. Ando. Applying alternating structure optimization to word sense disambiguation. In *(CoNLL-X)*, 2006.

R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Mach. Learn.*, 73(3):243–272, 2008. ISSN 0885-6125.

F. Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML*, 2004.

A. R. Barron and T. M. Cover. Minimum complexity density estimation. *IEEE Transactions on Information Theory*, 37(4):1034–1054, 1991.

A. R. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–, 1998.

P. Bickel and K. Doksum. Mathematical statistics. 2001.

R. Caruana. Multitask learning. In *Machine Learning*, pages 41–75, 1997.

C. C Chang and C.J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

J. Chen and M. Palmer. Towards robust high performance word sense disambiguation of english verbs using rich linguistic features. In *IJCNLP*, pages 933–944, 2005.

J. Chen, A. I. Schein, L. H. Ungar, and M. Palmer. An empirical study of the behavior of active learning for word sense disambiguation. In *HLT-NAACL*, 2006.

T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 2006.

P. S. Dhillon and L. H. Ungar. Transfer learning, feature selection and word sense disambiguation. In *Annual Meeting of the Association of Computational Linguistics, (ACL)*, August 2009.

P. S. Dhillon, D. Foster, and L. H. Ungar. Efficient feature selection in the presence of multiple feature classes. In *International Conference on Data Mining (ICDM)*, pages 779–784, 2008.

P. S. Dhillon, B. Tomasik, D. Foster, and L. Ungar. Multi-task feature selection usindhillong the multiple inclusion criterion (mic). In *European Conference on Machine Learning (ECML)-PKDD*, Lecture Notes in Computer Science. Springer, September 2009.

P. S. Dhillon, D. Foster, and L. Ungar. Feature selection using multiple streams. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 13, 2010.

B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

R. Florian and D. Yarowsky. Modeling consensus: classifier combination for word sense disambiguation. In *EMNLP '02*, pages 25–32, 2002.

D. P. Foster and E. I. George. The risk inflation criterion for multiple regression. *The Annals of Statistics*, 22(4):1947–1975, 1994. ISSN 00905364.

P. Grünwald. A tutorial introduction to the minimum description length principle. In *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2005.

P. D. Grünwald. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press, 2007. ISBN 0262072815.

E. H. Hovy, M. P. Marcus, M. Palmer, L. A. Ramshaw, and R. M. Weischedel. Ontonotes: The 90% solution. In *HLT-NAACL*, 2006.

J. Huang, T. Zhang, and D. Metaxas. Learning with structured sparsity. In *ICML '09*, 2009.

L. Jacob, G. Obozinski, and J-P. Vert. Group lasso with overlap and graph lasso. In *ICML '09*, 2009.

T. Jebara. Multi-task feature and kernel selection for SVMs. In *Proceedings of the twenty-first international conference on Machine learning*. ACM New York, NY, USA, 2004.

V. Kandylas, S. P. Upham, and L. H. Ungar. Finding cohesive clusters for analyzing knowledge communities. In *ICDM*, pages 203–212, 2007.

K. Kipper, H. T. Dang, and M. Palmer. Class-based construction of a verb lexicon. In *AAAI/IAAI*, pages 691–696, 2000.

S.-I. Lee, V. Chatalbashev, D. Vickrey, and D. Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *ICML '07*, pages 489–496, 2007. ISBN 978-1-59593-793-3.

B. Levin. *English Verb Classes and Alternations*. University of Chicago Press, 1993.

D. Lin. Review of wordnet: an electronic lexical database by christiane fellbaum. the mit press 1998. *Comput. Linguist.*, 25(2):292–296, 1999. ISSN 0891-2017.

D. Lin, E. Pitler, D. P. Foster, and L. H. Ungar. In defense of ℓ_0. In *Workshop on Feature Selection,(ICML 2008)*, 2008.

H. Liu and J. Zhang. On the ℓ_1-ℓ_q regularized regression. Technical report, Carnegie Mellon University.

K. Lounici. Sup-norm convergence rate and sign concentration property of lasso and dantzig estimators, 2008.

L. Meier, S. van de Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society. Series B*, 70(1):53–71, 2008.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34:1436–1462, 2006.

G. Miller. Special issue, wordnet: An on-line lexical database. *International Journal of Lexicography*, 4(3), 1990.

V. K. et. al. Mootha. Pgc-1alpha-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes, http://www.broad.mit.edu/gsea/datasets.jsp. *Nature Genetics*, 34:267 – 73, 2003/07// 2003.

Y. Nardi and A. Rinaldo. On the asymptotic properties of the group lasso estimator for linear models, 2008.

BK Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24:227, 1995.

G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 2009.

E. O. Perlstein, D. M. Ruderfer, D. C. Roberts, S. L. Schreiber, and L. Kruglyak. Genetic basis of individual differences in the response to small-molecule drugs in yeast. *Nat Genet*, 39, 2007.

R. Raina, A. Y. Ng, and D. Koller. Constructing informative priors using transfer learning. In *ICML '06*, pages 713–720, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2.

A. Rakhlin. Transfer learning toolkit. *http://multitask.cs.berkeley.edu*, 2007.

A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. Simplemkl. *JMLR*, 9:2491–2521, 2008.

J. Rissanen. Hypothesis selection and testing by the mdl principle. *The Computer Journal*, 42:260–269, 1999.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983.

K. K. Schuler. Verbnet: A broad coverage, comprehensive verb lexicon. In *Ph.D. Thesis, Computer and Information Sciences, University of Pennsylvania*, June 2006.

G.: Schwartz. Estimating the dimensions of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

Joel A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Inform. Theory*, 50:2231–2242, 2004.

B.A. Turlach, W.N. Venables, and S.J. Wright. Simultaneous variable selection. *Technometrics*, 47(3):349–363, 2005.

L. J. et. al. van 't Veer. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415(6871):530–536, January 2002. ISSN 0028-0836.

M. J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using l1-constrained quadratic programming (lasso). *IEEE Trans. Inf. Theor.*, 55(5):2183–2202, 2009. ISSN 0018-9448.

Z. Wu and H. H. Zhou. Model selection and sharp asymptotic minimaxity. *Under Submission*, -(-):–, 2010. ISSN -.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, February 2006. ISSN 1369-7412.

T. Zhang. Adaptive forward-backward greedy algorithm for sparse learning with linear models. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1921–1928. 2009a.

T. Zhang. On the consistency of feature selection using greedy least squares regression. *Journal of Machine Learning Research (JMLR)*, 10:555–568, 2009b. ISSN 1532-4435.

T. Zhang. On the convergence of mdl density estimation. In *COLT*, pages 315–330, 2004.

P. Zhao and B. Yu. On model selection consistency of lasso. *J. Mach. Learn. Res.*, 7: 2541–2563, 2006. ISSN 1532-4435.

P. Zhao, G. Rocha, and B. Yu. Grouped and hierarchical model selection through composite absolute penalties. *Annals of Statistics*, 2008.

S. Zhou. Thresholding procedures for high dimensional variable selection and statistical estimation. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2304–2312. 2009.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal Of The Royal Statistical Society Series B*, 67(2):301–320, 2005.