2008 Eighth IEEE International Conference on Data Mining

Efficient Feature Selection in the Presence of Multiple Feature Classes

Paramveer S. Dhillon
CIS
University of Pennsylvania
pasingh@seas.upenn.edu

Dean Foster
Statistics
University of Pennsylvania
foster@wharton.upenn.edu

Lyle H. Ungar
CIS
University of Pennsylvania
ungar@cis.upenn.edu

Abstract

We present an information theoretic approach to feature selection when the data possesses feature classes. Feature classes are pervasive in real data. For example, in gene expression data, the genes which serve as features may be divided into classes based on their membership in gene families or pathways. When doing word sense disambiguation or named entity extraction, features fall into classes including adjacent words, their parts of speech, and the topic and venue of the document the word is in. When predictive features occur predominantly in a small number of feature classes, our information theoretic approach significantly improves feature selection. Experiments on real and synthetic data demonstrate substantial improvement in predictive accuracy over the standard L_0 penalty-based stepwise and streamwise feature selection methods as well as over Lasso and Elastic Nets, all of which are oblivious to the existence of feature classes.

1. Introduction

Features often come in classes or groups that partition the entire feature set into subsets or feature classes. For example in biological datasets the genes may be divided into different gene families or pathways. SNPs (Single Nucleotide Polymorphisms) may be grouped by positional proximity on the chromosome. When doing word sense disambiguation [1], one can use adjacent words, the parts of speech of adjacent words and the topic of the document the word is in as feature classes. Similarly, when classifying scientific papers, feature classes include words in the paper, the journal or conference of the paper, citations to and from the paper, the paper's authors and the institutes to which they are affiliated. When predicting the function of a protein, feature classes include what proteins bind to it, what proteins in the same or other species have similar sequence or structure. More generally, starting from any feature set, one can generate new features by using techniques like Principal Component Analysis (PCA), Non Negative Matrix Factorization (NNMF), transformations such as log or square root, and by including interaction terms (i.e., products of the original features). Each feature generation method produces a feature class.

These feature classes differ in size and in how dense they are in "beneficial" features i.e. the features which should be included in a predictive model. Standard feature selection methods such as stepwise linear regression, which uses an L_0 penalty on the feature weights, or methods like the lasso, which use an L_1 penalty on the feature weights, are oblivious to the existence of this obvious structure in data; they consider the entire feature set as belonging to a single equivalence class. This paper shows that feature selection can be improved if we account for the existence of multiple feature classes when doing feature selection. The basic intuition behind our approach is that some feature classes will have a higher fraction of features that are beneficial and give better predictive accuracy, so once we find such feature classes, then we can preferentially draw features from those feature classes. More precisely, when controlling the false discovery rate (the probability of adding a spurious feature), if all features are in a single equivalence class, then any feature is penalized by the cost of considering *all* the features, while if features are divided into classes, then that feature need only be penalized for the search over other features in the same class, plus a smaller penalty relating to the number of classes.

We present an approach based on information theory to do efficient feature selection in data with multiple feature classes. When predictive features occur predominantly in a small number of feature classes, which is generally the case in real data, our information theoretic approach provides a significantly improved feature selection criterion. Our Three Part Coding (TPC) scheme is a penalized likelihood method based on the Minimum Description Length Principle (MDL) [7], and it exploits the information about the existence of multiple feature classes in the data to construct an efficient coding scheme.

We present experimental results on synthetic and real

1550-4786/08 $25.00 © 2008 IEEE
DOI 10.1109/ICDM.2008.56

Word Sense Disambiguation (WSD) [1] and gene expression data [5]. Our TPC methods show substantial improvement in performance over standard L_0 penalty-based standard stepwise and streamwise methods, as well as over other popular feature selection methods like Lasso (L_1 penalty) [8] and Elastic Nets (both L_1 and L_2 penalty) [12].

The rest of the paper is organized as follows. Section 2 provides a brief background on various feature selection methods, and how our work relates to them. In Section 3 we formulate the Three Part Coding (TPC) scheme. We demonstrate our results on synthetic and real data in Section 4. In Section 5 we talk about related work and we conclude in Section 6.

2. Background: Feature Selection Methods

Standard feature selection methods for supervised learning assume a setting consisting of *n* observations and a fixed number of *p* candidate features. The goal of feature selection is to select the feature subset that will lead to a model with least prediction error on test set. For many prediction tasks only a small fraction of the total *p* features are beneficial, so good feature selection methods can give large improvement in predictive accuracy.

The state of the art feature selection methods use either L_0 or L_1 penalty on the coefficients. L_1 penalty methods such as Lasso [8] and its variants [9, 10], being convex, can be solved by optimization and give guaranteed optimal solutions [2]. On the other hand, L_0 penalty methods require an explicit search through the feature space as in stepwise, stagewise and streamwise regression [11].

Penalized likelihood methods are widely used for feature selection. They minimize a score which can be represented as:

$$Score = -2log(likelihood) + Fq \tag{1}$$

where *F* is a function designed to penalize model complexity, and *q* represents the number of features currently included in the model at a given point. The first term in the above equation represents a measure of the in-sample error given the model, while the second term is a model complexity penalty. There are many different functions *F* that are used in practice. The most common ones being AIC (Akaike Information Criterion; $F = 2$), BIC (Bayesian Information Criterion; $F = log(n)$) and RIC (Risk Inflation Criterion; $F = 2log(p)$) [4], also known as a Bonferroni penalty.

Our Three Part Coding (TPC) approach uses a Minimum Description Length (MDL) [7] based coding scheme, which we explain in the next section, to specify another penalized likelihood method.

3. Three Part Coding (TPC) scheme

In this section, after describing our notation, we present the TPC scheme and compare it with standard RIC coding [4]. The symbols used throughout the rest of this section are defined in the following Table 1:

Table 1. Symbols used and their definitions.

Symbol	Meaning
n	Number of observations
p	Number of candidate features
p^*	Number of beneficial features in the candidate feature set
q	Number of features currently included in the model
Q	Number of feature classes currently added in the model
K	Total number of feature classes
p_k	Total number of candidate features in the k^{th} feature class

All the above values are given by the data, except p^* which is unknown, and q and Q, which are determined by the search/optimization procedure.

As mentioned earlier, TPC is a penalized likelihood method based on the principle of Minimum Description Length (MDL) [7]. In MDL, both sender and receiver are assumed to know the feature matrix, *X*. The sender's goal is to construct a model which uses *X* to describe the response vector *Y*. This model is sent as a "message" to the receiver so that the receiver can reconstruct *Y*. The goal is to minimize the sum of the length of this message and the error in reconstructing *Y*, both measured in bits. This sum is called the *Total description length* (TDL).

Equation 1 shows the two parts of the TDL. Let S_E be the number of bits for encoding the residual errors given the models, and let S_M be the number of bits for encoding the model. Then the TDL can be written as:

$$S = S_E + S_M \tag{2}$$

When we evaluate a feature, we want to maximize the reduction of TDL incurred by adding this feature to our model. This change in description length is:

$$\Delta S = \Delta S_E - \Delta S_M \tag{3}$$

where $\Delta S_E \geq 0$ is the number of bits saved in describing residual error due to increase in the likelihood of the data given the new feature and $\Delta S_M > 0$ is the extra bits used for coding this new feature.

In TPC ΔS_M has three parts; the class of the feature being added, which feature in the class, and what is its coefficient. We describe each of these codings in the following subsections.

Coding Scheme for ΔS_E :

ΔS_E represents the increase in likelihood of the data by adding the new feature to the model. When doing linear regression, we assume a Gaussian model and hence have:

$$P(y|w,x) = likelihood \sim \exp\left(-\left(\frac{\sum_{i=1}^{n}(y_i - wx_i)^2}{2\sigma^2}\right)\right) \quad (4)$$

where *y* is the response, *x's* are the features, *w's* are the regression weights and σ^2 is the variance of the Gaussian noise.

Alternatively, we can write Equation 4 to represent log(likelihood) in bits.

$$-log_2(likelihood) \sim \frac{E^2}{2\sigma^2 \ln 2} \quad (5)$$

where E^2 is the SSE (Sum of Squared Errors).

Intuitively, ΔS_E corresponds to the increase in benefit by adding the new feature to the model. It is always non-negative; even a spurious feature cannot decrease the training data likelihood.

Coding Scheme for ΔS_M :

To describe ΔS_M, when a new feature is added to the model, we use a three part coding scheme. Let l_C be the number of bits needed to code the index of the "feature class" of the evaluated feature, let l_I be the number of bits used to code the index of the evaluated feature in that particular feature class, and let l_θ be the number of bits required to code the coefficient of the evaluated feature. Thus:

$$\Delta S_M = l_C + l_I + l_\theta \quad (6)$$

This coding, as specified below, is the source of the power of our approach. Intuitively, if a feature class has many good (beneficial) features then we can share the cost of coding l_C across the features and hence save many bits in coding, as each feature costs roughly $log(p_k)$ bits to code rather than $log(p)$ as required by the standard RIC penalty. Now we explain how to code each of the three terms on right hand side of Equation 6.

Code l_C: l_C represents the number of bits required to code the index of the feature class to which the evaluated feature belongs. When we are doing feature selection by using TPC, two cases can arise:

Case 1: The feature class of the feature being evaluated is not yet included in the model. In this case, we code l_C by using $log(K)$ bits, where *K* is the total number of feature classes in the data. From now on, we will denote l_C under this case as l_C^1 .

Case 2: The feature class of the feature being evaluated is already included in the model. In this case, we can save some bits by coding l_C using $log(Q)$ bits where *Q* is the number of feature classes included in the model till that point of time. (Think of keeping an indexed list of length Q of the feature classes that have been selected). This is where TPC wins over other methods, as we do not need to waste bits on coding the feature class if it is already in the model. We will call l_C under this case as l_C^2.

We can summarize the coding scheme for l_C as follows:

$$l_C = \begin{cases} log(K) & if\ the\ feature\ class\ is\ not\ in\ the\ model \\ log(Q) & if\ the\ feature\ class\ is\ already\ in\ the\ model \end{cases} \quad (7)$$

Code l_I: l_I represents the number of bits required to code the index of the feature within its feature class. We have a total of p_k features in the k^{th} feature class. We use an RIC-style coding to code l_I i.e. we use log(p_k) bits to code the index of the feature. (This is equivalent to the widely used Bonferroni penalty). Since we also code the coefficient of the feature l_θ (unlike standard RIC), we do not overfit even when the usual RIC assumption of $n << p_k$ is not valid.

$$l_I = log(p_k) \quad (8)$$

Code l_θ: This term corresponds to the number of bits required to code the value of the coefficient of each feature. We could use either AIC or the more conservative BIC criterion to code the coefficients. We use 2 bits for each coefficient, which is quite similar to the AIC criterion and is based on the approach given in [6]. Therefore:

$$l_\theta = 2 \quad (9)$$

3.1 Analysis of TPC Scheme

We now compare the TPC coding scheme with a standard coding scheme (abbreviated as SCS below) in which we use an RIC penalty for feature indexes and an AIC-like penalty (2 bits) for the coefficients of the features, as this is the form of standard feature selection setting that comes closest to TPC in theory and in performance.

The Total Cost in bits used by SCS to code the *q* selected features is:

$$\begin{aligned} TotalCost_{SCS} &= \overbrace{[qlog(p)]}^{RIC\ Penalty} + \overbrace{[2q]}^{Coefficients} \\ &= qlog(K) + qlog(\frac{p}{K}) + 2q \quad (10) \end{aligned}$$

The total cost used by TPC to code the same features is:

$$TotalCost_{TPC} = \overbrace{Qlog(K)}^{l_C^1} + \overbrace{(q-Q)log(Q)}^{l_C^2} + \overbrace{qlog(p_k)}^{l_I} + \overbrace{2q}^{l_\theta} \quad (11)$$

The savings in coding comes from the (q-Q) features that belonged to classes that were already in the model.

Case 1: All Classes are of uniform size: In this case, $\log(p_k)$ in the Equation 11 will be equal to $\log(\frac{p}{K})$, as the size of each feature class will be same and will be equal to $\frac{p}{K}$, where p is the total number of candidate features and K is the total number of feature classes. So, subtracting Eq. (11) from Eq. (10) we get:

$$\Delta TotalCost = (q-Q)log(\frac{K}{Q}) \quad (12)$$

Equation 12 shows that TPC gives substantial improvement over SCS when either one or both of the conditions $q \gg Q$ or $K \gg Q$ are true. In other words, TPC wins when there are more features q than feature classes Q included in the model (i.e., there are multiple features per class) or, a smaller fraction, Q/K, of the feature classes include selected features.

In short, the real performance gain of TPC occurs when all or most of the (beneficial) selected features lie in small number of feature classes. The best case would occur when all the (beneficial) selected features lie in one class and the worst case occurs when the beneficial features are uniformly distributed across all the feature classes. In real datasets, the scenarios that we encounter lie somewhere between the best and the worst case, so we can expect substantial performance gain by using TPC.

Case 2: Classes are of nonuniform size: In this case, much of the theory remains the same as in Case 1, except that $C \neq \frac{p}{K}$. Let $\frac{p}{K} = p_{avg}$, i.e., the average size of a feature class. Then equation 12 becomes:

$$\Delta TotalCost = (q-Q)log(\frac{K}{Q}) + \overbrace{qlog(\frac{p_{avg}}{p_k})}^{Term2} \quad (13)$$

Now, it can easily be inferred that $p_{avg} > p_k$ occurs in the case when the beneficial features are in feature classes whose size is less than the average size of a feature class. Intuitively, $p_{avg} = p_k$ occurs if the size of all the feature classes is same (which was Case 1), so the performance of TPC will be improved in this case compared to Case 1 if the beneficial features lie in small classes. The improvement in performance over Case 1 will be quite significant when the beneficial features lie in a small class i.e. C is small or there are very big classes with no beneficial features in them, in either case the contribution of Term 2 in Equation 13 will increase.

4. Experimental Results

In this section we demonstrate the results of the TPC scheme on synthetic and real datasets. For our experiments we use the TPC coding scheme with stepwise and streamwise feature selection and compare against standard stepwise regression with an RIC penalty, standard streamwise feature selection, Lasso [8] and Elastic Nets [12]. Stepwise feature selection makes multiple passes through the data and at each iteration adds the best feature in the model (i.e., the feature that has the maximum ΔS). It stops when no feature provides better ΔS than in the previous iteration. In streamwise feature selection, each feature is considered only once for addition to the model, and added if it gives significant reduction in penalized likelihood, or otherwise discarded and not examined again. It is greedier than stepwise regression, and works well when there are millions of candidate features.

For Lasso and Elastic Nets we used their standard LARS (Least Angle Regression) implementations [2]. 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 [12] we chose the value of λ_2 (the weight on the L_2 penalty term), as 10^{-6}.

To demonstrate the results on real datasets we used Word Sense Disambiguation (WSD) [1] and gene expression datasets [5]. As is shown below, the results were quite encouraging.

4.1 Evaluation on Synthetic Data

In order to demonstrate the theoretical aspects of our TPC coding scheme, we first tested it on two synthetic datasets. For both the datasets, 1,000 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 A" had 4 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 B" 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 2. The number of correct and spurious features selected and 10-fold Cross Validation (CV) RMS error averaged over 10 runs. A). Unequal class sizes, B). Uniform class sizes.

Method	# Avg. Features Selected				10-Fold CV Error	
	Correct		Spurious			
	A	B	A	B	A	B
SRIC	4.4	3.2	0.2	0.0	0.27	0.61
STPC	6.8	5.6	0.1	0.3	0.09	0.27
SSm	1.9	1.4	0.1	0.0	0.78	0.80
SmTPC	5.3	4.1	0.5	0.1	0.17	0.39
Lasso	5.2	4.3	2.2	1.8	0.22	0.41
EN	6.4	4.9	3.3	2.1	0.20	0.43

SRIC - Stepwise RIC , STPC - Stepwise TPC, SSm - Standard Streamwise, SmTPC - Streamwise TPC, EN - Elastic Nets

As can be seen from the results in Table 2, in both cases the TPC versions of both stepwise and streamwise regression outperform their respective standard regression counterparts, as well as outperforming lasso and elastic nets.

4.2 Evaluation on Real Datasets

In order to benchmark the real world performance of our TPC coding scheme, we chose two datasets pertaining to two diverse applications of feature selection methods, namely Natural Language Processing (NLP) and Gene Expression Analysis. More information regarding the data and the experimental results are given below.

Word Sense Disambiguation (WSD) Dataset: A WSD dataset consisting of six ambiguous verbs and a rich set of contextual features [1] 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 WSD data range from 120 to 839 observations, 6,245 to 20,474 features and 59 to 75 classes.

A sample feature vector, given below, shows typical features and their classes. In each case, the part of the feature before the underscore is the feature class. Classes included pos (part of speech of the verb), morph (verb morphology), sub (the subject of the verb), subjsyn (the wordnet synonym set labels of the subject), dobj (the direct object of the verb), dobjsyn (dobj's wordnet synsets), word-1, word-2, word+1, word+2 (the words 1 or 2 before the verb or 1 or 2 after) pos-1, pos-2, pos-3, pos-4 (the parts of speech of those words), bigrams of the words, and tp (the topics of the document).

The results for the WSD Dataset are presented in the Tables 3 and 4. The standard errors for the different verbs and methods range from 0.0008 to 0.005, but TPC is always significantly better than the competing methods.

Table 3. Average # of Features Selected

Dataset	SRIC	STPC	SSm	SmTPC	Lasso	EN
WSD	17	17.2	36.5	22.5	16.5	16.3
GSEA	1	2.4	8.6	2.8	1.4	6.8

Table 4. 10-fold CV RMS errors for the WSD Dataset

Dataset	SRIC	STPC	SSm	SmTPC	Lasso	EN
ADD	0.42	0.39	0.29	0.28	0.42	0.40
BEGIN	0.31	0.27	0.29	0.27	0.32	0.32
CALL	0.16	0.15	0.26	0.23	0.24	0.30
CARRY	0.29	0.26	0.30	0.28	0.37	0.30
DEVELOP	0.42	0.41	0.44	0.43	0.52	0.48
DRAW	0.23	0.20	0.27	0.26	0.24	0.25

Gene Set Enrichment Analysis (GSEA) Datasets: The second real datasets that we used for our experiments were gene expression datasets from GSEA [5]. We used five gene expression datasets, with sizes ranging from 32 to 50 observations, 10,056 to 15,056 features and 182 to 318 features. GSEA provides many different divisions of genes into classes; we used gene classes C1: Positional Gene Sets (based on cytogenetic band), C2: Curated Gene Sets (based on pathway databases). The results for these GSEA datasets are as shown in the Tables 3 and 5.

Table 5. 10 Fold CV Errors (RMS Value) for GSEA Datasets

Dataset	SRIC	STPC	SSm	SmTPC	Lasso	EN
Leukemia	0.43	0.41	1.92	0.45	0.71	0.71
Gender 1	0.24	0.21	0.27	0.26	0.73	0.73
Diabetes	0.53	0.51	0.71	0.58	0.59	0.66
Gender 2	0.26	0.21	0.28	0.27	0.90	0.84
P53	0.53	0.52	0.54	0.49	0.75	0.72

For these datasets, TPC methods also beat the standard methods. The standard errors for the different data sets and methods range from 0.001 to 0.01 (with higher uncertainties corresponding to the higher errors on the L_1 methods), but TPC is again always significantly better than the competing methods. Interestingly,the TPC methods selected substantially fewer features on average than the other methods, but still gave better performance. This is consistent with the predictions of Equation 12 in that although the number of features selected, q may be small, the number of classes, K, is quite large for the GSEA datasets.

5. Related Work

Other potential methods that do feature selection in settings similar to TPC are Group Lasso [10] and GSEA [5]. Group Lasso is quite similar to the Standard Lasso (L_1 penalty) that we used for our experiments, but it uses a penalty which lies between L_1 and L_2. It induces sparsity at the level of feature classes ("factors"), unlike Lasso which does it at the level of features. Moreover it forces the coefficients in the same group to have similar values, which is not always a good assumption. For example, the expression levels of genes in the same pathway tend to predict (or not predict) cancer, but often their coefficients are of different signs. The TPC scheme differs from Group Lasso as TPC, being an L_0 penalty, does not put any constraints on the values of coefficients in the same class.

Gene Set Enrichment Analysis (GSEA) is another method that has become popular for analyzing gene expression data. GSEA (and its variants and extensions [3]) finds changes in expression levels of sets (classes) of genes selected *a priori* in transcriptional profiling experiments. Its goal is hypothesis testing: selecting *all* the features which might be correlated with Y. In contrast to this, TPC builds a predictive model and selects a *minimal* feature set. GSEA focuses more on the question **'Which feature classes are significant?'** in the case where there may be large numbers of overlapping feature classes, while TPC, concentrates on the question **'Which features are significant?'**

6. Concluding Remarks

In this paper we proposed a Three Part Coding (TPC) scheme for feature selection that is based on the principle of Minimum Description Length (MDL). As predicted by theory, TPC gives significant improvement in performance over standard feature selection methods when beneficial features are unevenly distributed across feature classes.

Although many data sets are presented as if they were a single undifferentiated matrix with a single type of feature, many problems do have natural feature classes. Words can be grouped by part of speech, by whether they are unigrams or bigrams, and by what dictionaries or word lists they appear on. Features of images can be grouped by their scale or by the method of generating them (e.g. wavelet or filter type or size). Metadata about objects often provides further classes of features: who wrote it, where was it published, when was it created or last modified, who has accessed it, etc. Such feature classes are almost universally present for documents such as web pages.

This paper has focused on settings where the problem domain suggests feature classes. TPC works equally well when features are derived by transformations of a single original feature set. Transformations such as computing principle components or generating interaction terms are particularly well-suited for the use of TPC, as the sizes of these feature classes are often widely divergent (10,000 features might generate 500 principle components, but 500 million interaction terms), and beneficial features are often concentrated in the smaller feature classes. (The 500 PCAs are more likely to contain highly predictive features, or at least ones that can be found to be significant, than the 500 million interaction terms.) When doing feature selection, it is important not to unduly penalize the 500 PCAs for the fact that one also wants to look at the 500 million interaction terms. Using an appropriate penalty term avoids precisely this problem.

References

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

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

[3] B. Efron and R. Tibshirani. On testing the significance of sets of genes. *Annals of Applied Statistics*, 1:107, 2007.

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

[5] V. K. Mootha, C. M. Lindgren, K.-F. Eriksson, A. Subramanian, S. Sihag, J. Lehar, P. Puigserver, E. Carlsson, and Ridderstr. Pgc-1alpha-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. *Nature Genetics*, 34:267 – 73, 2003/07// 2003.

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

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

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

[9] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B*, 67(1):91–108, 2005.

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

[11] J. Zhou, D. P. Foster, R. A. Stine, and L. H. Ungar. Streamwise feature selection. *Journal of Machine Learning Research*, 7:1861–1885, September 2006.

[12] 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.