Streaming Feature Selection using IIC

Lyle H. Ungar and Jing Zhou
Computer and Information Science
University of Pennsylvania, Philadelphia, PA 19104
ungar, jingzhou@seas.upenn.edu

Dean P. Foster and Bob A. Stine
Statistics Department
University of Pennsylvania, Philadelphia, PA 19104
foster, stine@wharton.upenn.edu

Abstract

In Streaming Feature Selection (SFS), new features are sequentially considered for addition to a predictive model. When the space of potential features is large, SFS offers many advantages over methods in which all features are assumed to be known in advance. Features can be generated dynamically, focusing the search for new features on promising subspaces, and overfitting can be controlled by dynamically adjusting the threshold for adding features to the model. We present a new, adaptive complexity penalty, the Information Investing Criterion (IIC), which uses an efficient coding of features added, and not added, to the model to dynamically adjust the threshold on the entropy reduction required for adding a new feature. Streaming Feature Selection with IIC gives strong guarantees against overfitting. In contrast, standard penalty methods such as BIC or RIC always drastically over- or under-fit in the limit of infinite numbers of non-predictive features. Empirical results show that SFS is competitive with much more compute-intensive feature selection methods.

1 Introduction

In many problems, one has a fixed set of observations from which a vast, or even infinite stream of features can be generated to build predictive models. The large number of potentially predictive features may come from transformations of, and interactions between, a smaller initial set of features. For example, most commercial statistical software offers the ability to do stepwise regression using all feature interactions (e.g., products of pairs of features, or all products containing three variables). Pairwise interactions are important and, along with data transformations, can rapidly create large data sets. For example in a bankruptcy prediction problem described below, considering interactions between the 365 original features led to a set of over 67,000 potential features, of which about 40 proved significant.

The features may also come from more complex feature generation algorithms. For example, Statistical Relational Learning (SRL) methods often generate tens or hundreds of thousands of potentially predictive features. SRL and related methods "crawl" through a database or other relational structure and generate features by building increasingly complex compound relations [1] . For example, when building a model to predict the journal in which an article will be published, potentially predictive features include the words in the target article itself, the words in the articles cited by the target article, the words in articles that cite articles written by the authors of the target article, and so forth. Traversing such relational structures can easily generates millions of features, since there are many words, authors, and journals. Current modeling techniques, however, are ill equipped to deal with problems of learning from, say, a million potential features for each of a hundred thousand observations. A hundred billion numbers do not fit easily into memory on most contemporary computers. More importantly, CPU is fast relative to memory, and being more so.

When building models from potentially enormous sets of features, it is desirable to interleave the process of feature generation with that of feature testing in order to avoid even generating features which are less likely to be useful. One may want to only consider interaction terms in a regression if at least one of the component terms has proven predictive. One may want to only search farther in those branches of a refinement graph in inductive logic programming (ILP) which contain terms that have proven predictive – as is, indeed, done in ILP. Building predictive models from such large, complex data sets requires careful control to avoid over-fitting, particularly when there are many more features than observations. Standard statistical and machine learning methods such as SVMs, maximum entropy methods and neural networks generally assume that all features ("predictors") are known in advance. They then use regu-

larization or features selection to avoid overfitting.

This paper focuses on penalty-based feature selection methods for problems in which a small number of predictive features are to be selected from a large set of potential features. We will compare, in the context of streaming feature selection, the widely used BIC penalty method with RIC, a more recent penalty method, and with the new Information Investing Criterion (IIC), which this paper introduces.

BIC can be understood in an information theoretic sense as consisting of a code (specifying the parameters in the model) and the compressed data (describing the errors in the predictions made by the model). Each zero parameter (feature not included in the model) is coded with one bit, and each non-zero parameter is coded with $1 + \frac{1}{2}\log(n)$ bits, where n is the number of observations used. (All logs are base 2.) Recalling that the log likelihood of the data given a model gives the number of bits to code the model error, leads to the BIC criterion for feature selection: accept a new feature x_i only if the change in log likelihood from adding the feature is greater than $\frac{1}{2}\log(n)$, i.e. if $\log(P(Y|\hat{Y}_i)) - \log(P(Y|\hat{Y}_{-i})) > \frac{1}{2}\log(n)$. BIC is equivalent to a Minimum Description Length (MDL)[2] criterion if the number of features considered, p is much less than the number of observations, n. Howver, BIC is not a valid code for $p \gg n$.

The Risk Inflation Criterion (RIC) [3, 4] gives another, much more stringent criterion for feature selection, which controls the minimax risk. RIC chooses a set of features from the potential feature pool so that the loss of the resulting model is within a factor of $\log(p)$ of the loss of the best such model. In essence, RIC behaves like a Bonferroni rule, in which a threshold for feature inclusion is selected so that *the set of all features* will only have a small chance of containing a "false" feature. This is highly conservative, and does often not produce optimal out of sample prediction accuracies.

The Information Investing Criterion (IIC) introduced in this paper is an alternative MDL-style coding which, unlike BIC and RIC, is adaptive. Information investing does not require knowing the number of potential predictors in advance, yet still has provable bounds on overfitting. IIC's performance is never much worse than BIC or RIC, and for the types of problems we are interested in, where there are far more potential features than observations, it often gives vastly superior performance.

The assumptions behind penalty methods such as BIC and RIC are not met when a fixed number of features are to be selected from an arbitrarily large set of potentially predictive features. Inclusion rules such as AIC and BIC, which are not a function of p, the number of possible features to be considered for inclusion in the model, inevitably overfit as p becomes large. When presented with a continuous sequence of features that are random noise, any selection procedure that generates false positives at a fixed rate, such as AIC or BIC, will select infinitely many of these random features as predictors. Inclusion rules such as RIC (Bonferroni) which *are* a function of p under-fit as p becomes large. Any such method that reduces the chance of including each feature based on the total number of features, p, to be considered will end up not adding any features in the limit as $p \to \infty$.

The solution to this dilemma is to sequentially consider a $stream$ of features for model inclusion and use a method which incrementally adjusts the criterion for including new features in the model depending on the history of addition (or non-addition) of features seen so far. We argue for Streaming Feature Selection (SFS), where as each additional feature is observed, it is tested for inclusion in the model and then either included or discarded. Streaming feature selection offers many advantages over the traditional approach of stepwise selection from a fixed set of features. In stepwise regression, all features are considered for addition to the model, the best one is selected, and then all remaining features considered, etc. At every iteration, almost all features are tested for addition to the model. This requires having a finite set of features specified in advance, and requires looking at each feature many times. Stepwise feature selection is widely used with penalty methods such as AIC and BIC, but we will show below that streaming feature selection often gives competitive performance, while allowing much greater flexibility in dynamically controlling feature generation. Using streams of features has other benefits. Since most features will not be included in the models, they can be discarded soon after generation, thus reducing data storage requirement and allowing the solution of larger problems than can be tackled using standard machine learning algorithms such as support vector machines (SVMs) or neural networks which assume that all potentially predictive features are known *a priori*.

2 Streaming feature selection

The goal of streaming feature selection is to pick useful predictors from an offered sequence of features. For a fixed set of observations, new features (predictors) are considered sequentially, and the decision to include or discard each feature is made at the time it is provided. SFS can be used with a variety of different machine learning methods; all it requires from the machine learner is that it take features sequentially and produce an estimate of the change in entropy (log-likelihood) in the model. A wide range of classical statistical methods can be used off-the-shelf, such as linear or logistic regression, or extensions such as generalized linear methods and estimating equations. SFS works particularly well with modeling methods than can efficiently add additional features and with adaptive penalty methods such as IIC.

$Initialize$
  $i = 1, wealth = w_0$ bits, $model = \{\}$
$Do\ forever$
  $x \leftarrow get\_new\_feature()$
  $\epsilon \leftarrow wealth/2i$
  $bits\_saved \leftarrow entropy\_reduction(x, \epsilon, model)$
  $if(bits\_saved > w_\Delta)$
    $wealth \leftarrow wealth + w_\Delta$
    $add\_feature(x, model)$ // add x to the model
  $else$
    $wealth \leftarrow wealth - \epsilon$
  $i \leftarrow i + 1$

Figure 1: Information-investing algorithm

SFS dynamically adjusts the threshold, w_Δ, on the entropy reduction needed for a new variable to enter the model.[1] The threshold, w_Δ, is adjusted using the wealth, w_i, which represents the number of bits currently available for overfitting. Wealth starts at an initial value w_0 specifying the number of bits by which one is willing to risk increasing the description length. It is increased by w_Δ each time a variable (feature) is added to the model, since the variable is guaranteed to save at least w_Δ bits, and decreased by ϵ each time a variable is not added to the model,reflecting (as described below) the cost of coding the fact that the feature was not added.

The algorithm is given in Figure 1. ϵ specifies how many bits are available to code a variable. The $bits\_saved$ by adding a feature to the model is the net entropy reduction from adding x to the model: the reduction in the model error minus the cost of coding the coefficient, β, associated with x and the cost of indicating that the variable is to be added to the model. Different codings can be used for the coefficients, for example $\frac{1}{2}log(n)$ bits (or, for a very approximate coding 3 bits) to code each nonzero coefficient, and e.g. -$\log(\epsilon)$ bits to code that x is to be added to the model. (Since ϵ is the number of bits available to code a spurious feature, the probability of the next feature being "useful." is $1 - e^{-\epsilon} = 1 - (1 - \epsilon + O(\epsilon^2)) \approx \epsilon$, and the cost in bits of coding that that the feature is useful is roughly -$\log(\epsilon)$ bits.) If β, the coefficient of x, has an associated t-statistic, then adding x to the model reduces the entropy of the model by $\frac{1}{2}t^2log(e)$. (The log(e) converts the t^2 to bits.)

If the feature x reduces entropy sufficiently to be worth adding to the model, then the wealth is incremented by a fixed amount, w_Δ. If the feature x is not added to the model, the wealth is decreased by the cost of coding the variable's absence, which by an argument similar for that used above is $-\log(1-\epsilon)$, which, for small ϵ, is approximately ϵ.

[1] A very similar SFS algorithm, which we call α-$investing$, can be written that dynamically adjusts the criterion for adding a new feature to a model based on the p-value of the feature under consideration.

2.1 Guarantees against overfitting

One sense in which SFS is guaranteed not to over-fit, is that on average, the sum of the total description length plus the wealth will never increase. Since the wealth is strictly positive, this guarantees that the total description length can never increase by more than the current wealth. Since when a feature is added to the model we increase the wealth less than the description length decreases, the description length plus wealth tends to decrease, providing on average better models.

SFS also provides another, much more subtle, guarantee against overfitting. For the case of "hard" problems, where the coefficients to be estimated are just barely distinguishable above the noise, the cost of adding a "false" feature is comparable to the benefit of adding a true features. This is a property of using a so-called *testimator*. A testimator tests for significance and then estimates by the usual estimator if it is significant, and estimates by zero otherwise. If a variable has a true coefficient of zero, then when it is falsely included, it will be biased by about $t_\alpha SE$, where t_α is the critical value used for testing significance, and SE is the standard error of the coefficient. On the other hand, the hardest to detect coefficients will have a coefficient of about $t_\alpha SE$. Hence leaving them out will bias their estimated value by about the same amount, namely $t_\alpha SE$. We can thus get optimal test error by adding as many features as possible while not exceeding a specified ratio of false to true features added to the model.[2]

SFS using the IIC coding (described below) allows us, for any valid coding, to bound in expectation the ratio of incorrect features added to correct features added, and thus to minimize the expected test error by adding as many features as possible subject to controlling that ratio.

Theorem
Let M_i be the number of correct variables included in the model, let N_i be the number of spurious variables (those with true coefficient zero) included and w_i be the wealth, all at iteration i, and let $w_\Delta < 1/4$ be a user selected value. Then if the algorithm in Figure 1 is modified so that it never bids more than $1/2$ it will have the property that:

$$E(N_i) < 4w_\Delta E(M_i) + 4w_0.$$

[2] This is very similar to controlling the False Discovery Rate (FDR) [5], the number of features incorrectly included in the model divided by the total number of features included in the model, which has become popular in recent years. In the regime that we are working, correctly adding a feature always reduces both the FDR and the out-of-sample error, and incorrectly adding a feature always increases both FDR and error.

Proof Sketch
The proof relies on the fact that $S_i \equiv N_i - 4w_\Delta M_i + 4w_i$ is a super-martingale, namely at each time period the conditional expectation of $S_i - S_{i-1}$ is negative. We will show that S_i is a super-martingale by considering the cases when the variable is or is not in the true model and is or is not added to the estimated model.

	$\beta_i = 0$	$\beta_i \neq 0$
use zero	$\Delta M_i = 0, \Delta N_i = 0$	$\Delta M_i = 0, \Delta N_i = 0$
add variable	$\Delta M_i = 0, \Delta N_i = 1$	$\Delta M_i = 1, \Delta N_i = 0$

We can write the change in S_i as:

$$\begin{aligned} \Delta S_i &\equiv S_i - S_{i-1} \\ &= \Delta N_i - 4w_\Delta \Delta M_i + 4\Delta w_i \end{aligned}$$

If $\beta_i \neq 0$, then $\Delta N_i = 0$. Thus,

$$\Delta S_i = -\epsilon_i(1 - \Delta M_i) \leq 0,$$

where ϵ_i is the amount bid at time i. On the other hand, if $\beta_i = 0$, then $\Delta M_i = 0$. So,

$$\begin{aligned} \Delta S_i &= \Delta N_i + 4\Delta w_i \\ &= \Delta N_i + 4w_\Delta \Delta N_i - 4\epsilon_i(1 - \Delta N_i) \\ &= \Delta N_i(1 + 4w_\Delta + \epsilon_i) - 4\epsilon_i. \end{aligned}$$

By bounds from information theory, we see that $E(\Delta N_i) \leq \epsilon_i$. Also by assumption, $4w_\Delta \leq 1$ and $\epsilon_i \leq 1/2$. Hence $E(\Delta S_i) \leq 4\epsilon_i - 4\epsilon_i = 0$. Thus, S_i is a super-martingale.

Using the weaker fact that for super-martingales: $E(S_i) \leq E(S_{i-1})$, we see that $E(S_i) \leq S_0$. But since we start out with $N_i = 0$, and $M_i = 0$, $S_0 = 4w_0$. Since $w_i > 0$ by construction, we see that $E(N_i - 4w_\Delta M_i) < 4w_0$. □

When $w_\Delta = \frac{1}{4}$, this reduces to $E(N_i) < E(M_i) + 4w_0$. The expected number of spurious variables added is thus no more than $4w_0$ greater than the expected number of correct variables added.

As described above, if we add as many features as possible subject to meeting such a constraint on spurious to true features added, we will minimize the expected test error. The selection of ϵ_i as $w_i/2i$ gives the slowest possible decrease in wealth such that all wealth is used; i.e., so that as many features as possible are included in the model without systematically over-fitting. More formally:

Theorem
Computing ϵ_i as $w_i/2i$ gives the slowest possible decrease in wealth such that $\lim_{i\to\infty} w_i = 0$.

Proof Sketch
Define $\delta_i = \epsilon_i/w_i$ to be the fraction of wealth invested at time i. If no features are added to the model, wealth at time i is $w_i = \Pi_i(1 - \delta_i)$. If we pass to the limit to generate w_∞, we have $w_\infty = \Pi_i(1 - \delta_i) = e^{\sum \log(1-\delta_i)} = e^{-\sum \delta_i + O(\delta_i^2)}$. Thus, $w_\infty = 0$ iff $\sum \delta_i$ is infinite.

Thus if we let δ_i go to zero faster than $1/i$, say $i^{-1-\gamma}$ where $\gamma > 0$ then $w_\infty > 0$ and we have wealth that we never use.

2.2 IIC and its coding scheme

A key question is what coding scheme to use in determining the entropy reduction. We describe here an "optimal" coding scheme which leads to the information investing algorithm described in Figure 1. Our goal is to find a (legitimate) coding scheme which, given a "bid," ϵ, specifying how many bits are available to code a variable, will guarantee the highest probability of adding the variable to the model. The key idea is that log(probability) and bits are equivalent. This equivalence allows us to think in terms of distributions and thus to compute codes which handle fractions of a bit. We show in this section that given any actual distribution $\tilde{f}_\beta$ of the coefficients, we can produce a coding corresponding to a modified distribution f_β which uniformly dominates the coding implied by $\tilde{f}_\beta$.

Assume, for simplicity, that we increase the wealth by one bit when a variable x_i with coefficient β_i is added. Thus, when x_i is added $\log(p(x_i \text{ is a "true" variable}) / p(x_i \text{ is a "false" variable})) > 1$ bit; i.e. the log-likelihood decreases by more than one bit. Let f_{β_i} be the distribution implied by the coding scheme for t_{β_i} if we add x_i and $f_0(t_{\beta_i})$ be the normal distribution (the null model in which x_i should not be added). The coding saves enough bits to justify adding a variable whenever $f_{\beta_i}(t_{\beta_i}) \geq 2 * f_0(t_{\beta_i})$. This happens with probability $\alpha_i \equiv p_0(\{t_{\beta_i} : f_{\beta_i}(t_{\beta_i}) \geq 2 * f_0(t_{\beta_i})\})$ under the null (α_i is thus the area under the tails of the null distribution.)

There is no reason to have $f_{\beta_i}(t_{\beta_i}) \gg 2 * f_0(t_{\beta_i})$ in the tails, since this would "waste" probability or bits. Hence the optimal coding corresponds to $f_\beta(t_{\beta_i}) = 2 * f_0(t_{\beta_i})$ for all the variables that are likely to be added. Using all of the remaining probability mass (or equivalently, making the coding "Kraft tight") dictates the coding for the case when the variable is not likely to be added. The most efficient coding to use is thus:

$$\begin{cases} f_\beta(t_{\beta_i}) = 2f_0(t_{\beta_i}) & \text{if } |t_{\beta_i}| > t_{\alpha_i} \\ f_\beta(t_{\beta_i}) = \frac{1-2*\alpha_i}{1-\alpha_i} f_0(t_{\beta_i}) & \text{otherwise} \end{cases}$$

and the corresponding cost in bits is:

$$\begin{cases} \log(f_\beta(t_{\beta_i})/f_0(t_{\beta_i})) = \log(2) = 1 \text{ bit} & \text{if } |t_{\beta_i}| > t_{\alpha_i} \\ \log(f_\beta(t_{\beta_i})/f_0(t_{\beta_i}) = & \\ \qquad \log(\frac{1-2\alpha_i}{1-\alpha_i}) \approx -\alpha_i \text{ bits} & \text{otherwise} \end{cases}$$

Figure 2 shows the distribution $f_\beta(t_(\beta_i))$, with the probability mass transfered away from the center, where features are not added, out to the tails, where features are added.

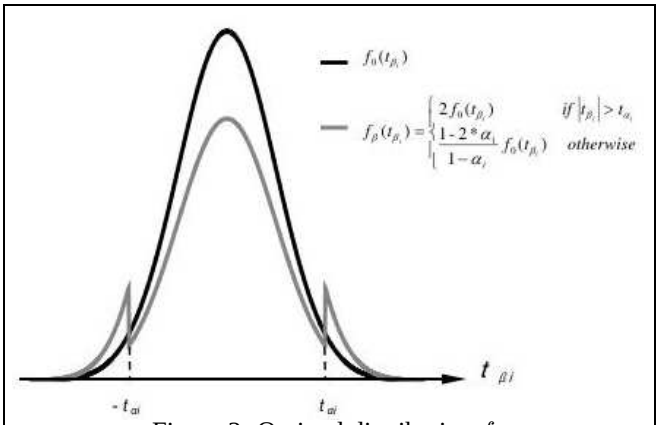

Figure 2: Optimal distribution f_β

		BIC	RIC	SFS
streaming	**features**	39.3	7.1	5.4
	error	3.21	2.88	3.16
stepwise	**features**	199	11.1	–
	error	3.89	2.40	–

Table 1. BIC overfits for $p \gg n$. Average number of features selected and out-of-sample error. n = 200 observations, p = 1,000 features, q = 10 true features in model Synthetic data: $x \sim N(0,1)$ y: linear in x with noise $\sigma^2 = 5$. A perfect model would give test error of 2.236, the error of the null model is 3.873. The results are an average over 20 runs, and reported errors have an uncertainty of around 0.02.

The above equations been derived assuming that 1 bit is added to the wealth. It can be generalized to add w_Δ bits to the wealth each time a variable is added to the model. Then, when a variable is added to the model the probability of it being "true" should be 2^{w_Δ} times that of it being "false", and all of the 2's in the above equations are replaced with 2^{w_Δ}.

3 Experimental Results

To further illustrate the method, we evaluate SFS on a synthetic data set for which the correct answers are known and on a larger, real data set of bankruptcy prediction. The base synthetic data set contains 200 observations each of 1,000 features, of which 10 are predictive. We generate the features independently from a normal distribution, $N(0,1)$, with the true model being the sum of the ten predictors plus noise, $N(0,5)$. The artificially simple structure of the data allows us to easily see which feature selection methods are adding spurious variables or failing to find variables that should be in the model.

The results are presented in Table 1. As expected, BIC overfits, although less badly when streaming is used rather than the stepwise selection procedure. RIC gives performance superior to SFS in this particular case ($q = 10$) but it fails badly when its assumptions (q small) are violated, as shown in Table 2. Stepwise regression using RIC does better here than the streaming version. However, using standard code from R, the stepwise regression was *much* slower than the streaming regression, to the point where running stepwise regression on data sets with tens of thousands of features was not possible.

	BIC	RIC	SFS
features	89.3	25.5	61.5
error	6.24	9.57	7.60

Table 2. RIC underfits for $q \gg 1$. Same parameters as Table 1 (streaming) except n = 1,000, q = 100 features in data and $\sigma^2 = 15$.

One might hope that adding more spurious features to the end of a feature stream would not severely harm an algorithm's performance. However, BIC, since its penalty is not a function of p, will add even more spurious variables (if BIC haven't already added a feature for every observation!). RIC (or Bonferroni) puts a harsher penalty as p gets large, adding fewer and fewer features. As Table 3 shows, SFS is clearly the superior method when the true features occur early in the feature stream. SFS continues to occasionally add features to the model, which would be good if there were predictive features later in the stream, but does not lead to much overfitting when there are no such features.

It is often the case that large numbers of features are generated, with the best ones tending to be earlier in the sequence. Such feature streams are generated when one searches over interactions and transformations, as in the bankruptcy example presented below. Similar feature streams arise when one computes features at many length scales, as for face or object recognition in machine vision. Another example is Structural Relational Learning (SRL), where potential features are generated by searching the space of logic queries or relational database queries.

We have used the CiteSeer data set, which contains about 500,000 papers, 100,000 "constants" (words, authors, and journals), and around ten different relations (including author, venue, cites, has-word, institution, download) to predict which journal a given paper will be published in, or which papers it will cite. Predictions using CiteSeer benefit from the generation of rich sets of features, and depending on the exact task, SFS gives out-of-sample errors comparable to, or several percentage points below those from non-adaptive techniques [6]. Learning in SRL methods such as Structural Generalized Linear Regression (SGLR) [6] benefit from efficient integration of feature generation and selection; as each feature is tested for possible inclusion in the model, the results are fed back to the feature genera-

	p	1,000	10,000	100k	1M
BIC	**features**	39.3	199	199	199
	false pos.	29.5	189	189	189
	error	3.21	4.45	4.45	4.45
RIC	**features**	7.1	3.8	1.2	0.9
	false pos.	0.1	0.5	0.1	0.2
	error	2.88	3.49	3.77	3.91
SFS	**features**	5.4	5.4	5.7	5.7
	false pos.	0.3	0.5	0.8	0.8
	error	3.16	3.30	3.29	3.29

Table 3. Effect of adding spurious features Same parameters as Table 1 except that additional spurious features have been added after the first 1,000 features. "false pos." indicates the average number of features incorrectly added. (average over 10 runs)

tor, which can then use this information to determine which further features to generate. Since generating the features from these databases takes CPU days, avoiding generating features is important both for computational as well as for statistical efficiency methods.

We also tested a slight modification of SFS on a problem of predicting personal bankruptcies[7]. The data set is highly un-balanced, containing 2,244 bankruptcy events and hundreds of thousands of non-bankruptcy observations. The real world loss function for predicting bankruptcy is quite asymmetric: the cost of predicting a bankruptcy when none occurs is much higher than the cost of failing to predict a bankruptcy when one does occur. We call the ratio of these two costs ρ.

We compared Streaming Feature Selection against boosted C4.5, doing 5-fold cross-validation, where each pass of the cross-validation uses 100,000 non-bankruptcies and about one fifth of the bankruptcies. SFS was run once, and then the out-of-sample costs were estimated for each cost ratio, ρ using the predicted probability of bankruptcy. C4.5 was run separately for each value of ρ.

ρ	199	99	19	6	4	1
C.45 cost	132	76	18.6	7.2	5.09	1.45
SFS cost	61	41	15.3	6.9	5.02	1.54

Table 4. Loss as a function of the loss ratio, ρ, for boosted C4.5 and for SFS

Table 4 shows that for low cost ratios, the two methods give very similar results, but at higher cost ratios, SFS gives around half the loss of C4.5. Using AIC, one would expect over 1,000 variables to be falsely included in the model, based on the fact that an f-statistic-based penalty of 2 corresponds to a t-statistic of $\sqrt{2}$ which is a wildly generous threshold when considering 67,000 features. BIC also massively overfits, although less severely.

4 Alternate feature selection methods

Recent developments in statistical variable selection take into account the size of the feature space, but only allow for finite, fixed feature spaces, and do not support sequential (or streaming) feature selection. The risk inflation criterion (RIC) produces a model that possesses a type of competitive predictive optimality [4, 3]. RIC chooses a set of features from the potential feature pool so that the loss of the resulting model is within a factor of $\log(p)$ of the loss of the best such model. In essence, RIC behaves like a Bonferroni rule [3]. Each time a predictor is considered, there is a chance that it will enter the model even if it is merely noise. In other words, the tested null hypothesis is that the proposed feature does not improve the prediction of the model. Doing a formal test generates a p-value for this null hypothesis. Suppose we only add this predictor if its p-value is less than α_j when we consider the jth predictor. Then the Bonferroni rule keeps the chance of adding even one extraneous predictor to less than, say, 0.05 by constraining $\sum \alpha_j \leq 0.05$.

Bonferroni methods like RIC are conservative, limiting the ability of a model to add factors that improve its predictive accuracy. The connection of RIC to α-spending rules leads to a more powerful alternative. An α-spending rule is a multiple comparison procedure that bounds its cumulative type 1 error rate at a small level, say 5%. For example, suppose one has to test the p hypotheses $H_1, H_2, \ldots, H_p$. If we test the first using level α_Δ, the second using level α_2 and so forth with $\sum_j \alpha_j = 0.05$, then we have only a 5% chance of falsely rejecting one of the p hypotheses. If we associate each hypothesis with the claim that a predictor adds to value to a regression, then we are back in the situation of a Bonferroni rule for variable selection. Bonferroni methods and RIC simply fix $\alpha_j = \alpha/p$ for each test.

Alternative multiple comparison procedures control a different property. Rather than control the cumulative α (also known as the family wide error rate), these control the so-called false discovery rate [5]. Control of the false discovery rate at 5% implies that at most 5% of the rejected hypotheses are false positives. In variable selection, this implies that of the included predictors, at most 5% degrade the accuracy of the model. The Benjamini-Hochberg method for controlling the false discovery rate suggests the α-spending method for keeping the false discovery rate below α: Order the p-values of the independents tests of $H_1, H_2, \ldots, H_p$ so that $p_1 \leq p_2 \leq \cdots p_p$. Now find the largest p-value for which $p_k \leq \alpha/(p-k)$ and reject all H_i for $i \leq k$. Thus, if the smallest p-value $p_1 \leq \alpha/p$, it is rejected. Rather than compare the second largest p-value to the RIC/Bonferroni threshold α/p, reject H_2 if $p_2 \leq 2\alpha/p$.

There have been many papers that looked at procedures of this sort for use in variable selection from an FDR perspective [8], an empirical Bayesian perspective [9, 10], an information theoretical perspective [11] or simply a data mining perspective [7]. But all of these require knowing the entire list of possible variables ahead of time. Further, most of them assume that the variables are orthogonal and hence tacitly assume that $p < n$.

We are currently exploring a way of doing SFS that uses what we call α-investing instead of IIC. In SFS using IIC, we keep track of the number of bits saved and use these bits to invest in future variables. Whereas in α-investing the medium of exchange is the accumulation of α that has yet to be spent. When a significant variable is found, the α-spending account goes up, but when a variable is found to be insignificant, the account decreases. Though the α-investing rule sounds like it might be close to Benjamini-Hochberg's FDR procedure described above, it turns out to be fairly different. In particular, the Benjamini-Hochberg method fails as p gets large; it is a batch-oriented procedure. But the α-investing shares with IIC the property of not needing to know p ahead of time and hence being able to handle a potentially infinite stream of predictors.

5 Summary

A variety of machine learning algorithms have been developed for online learning where *observations* are sequentially added. Algorithms such as SFS which are online in the *features* being used are much less common. For some problems, all predictors are known in advance, and a large fraction of them are predictive. In such cases, regularization or smoothing methods work well and streaming feature selection does not make sense. For other problems, selecting a small number of features gives a much stronger model than trying to smooth across all potential features. (See [12, 13] for a range of feature selection problems and approaches.) For example, in predicting what journal an article will be published in, we find that roughly 10-20 of the 80,000 features we examine are selected [14]. For the problems in citation prediction and bankruptcy prediction that we have looked at, generating potential features (e.g. by querying a database or by computing transformations or combinations of the raw features) takes orders of magnitude more time than the machine learning done by streaming feature selection. Thus, the flexibility that SFS provides to dynamically decide which features to generate and add to the feature stream provides potentially large savings in computation.

Streaming feature selection can be done using any penalty method such as AIC, BIC or RIC, but is functions best when using a method such as IIC which adapts the penalty as a function of what features have been seen and added at each point. The widely used BIC criterion is only valid in the limit as the number of observations n goes to infinity while the number of features p remains small. The more modern RIC assumes that n and p are large but that the number of true variables in the model is close to one. Unlike BIC and RIC, IIC works for all values of p and n, and for any $q \ll p$. The results presented in this paper are for "hard" problems, in which the coefficients are close to the limit of being detectable above the noise. For easy problems, where the signal to noise ratio is high, all methods tend to work reasonably well. For problems which have a mix of easy and hard coefficients, the SFS algorithm can be modified to make multiple passes, first "investing" a relatively small number of bits to find the easy features, and then using the algorithm as described above to find the hard features.

Key to the guarantee that IIC works for widely varying values of n, p and q is the use of an adaptive penalty to control the ratio of correct ("true") to incorrect ("false") features by using an information theoretic coding to adjust the threshold on the entropy reduction necessary for adding a variable to the model. Streaming Feature Selection with IIC is extremely easily to implement on top of any algorithm which incrementally considers features for addition and calculates their entropy reduction or p-value. For linear and logistic regression, we have found that SFS can easily handle millions of features.

References

[1] S. Dzeroski and N. Lavrac. *Relational Data Mining*. Springer-Verlag, 2001.

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

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

[4] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81:425–455, 1994.

[5] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society*, Series B(57):289–300, 1995.

[6] A. Popescul and L. H. Ungar. Cluster-based concept invention for statistical relational learning. In *Proc. Conference Knowledge Discovery and Data Mining (KDD-2004)*, 2004.

[7] D. P. Foster and R. A. Stine. Variable selection in data mining: Building a predictive model for bankruptcy. *Journal of the American Statistical Association (JASA)*, 2004. 303-313.

[8] Felix Abramovich, Y. Benjamini, D. Donoho, and Ian Johnstone. Adapting to unknown sparsity by controlling the false discovery rate. Technical Report 2000–19, Dept. of Statistics, Stanford University, Stanford, CA, 2000.

[9] E. I. George and D. P. Foster. Calibration and empirical bayes variable selection. *Biometrika*, 87:731–747, 2000.

[10] I. M. Johnstone and B. W. Silverman. Needles and straw in haystacks: Empirical bayes estimates of possibly sparse sequences. *Annals of Statistics*, 32:1594–1649, 2004.

[11] D. P. Foster and R. A. Stine. Adaptive variable selection competes with Bayes experts. Submitted for publication, 2004.

[12] In *JMLR Special Issue on Variable Selection*. Journal of Machine Learning Research (JMLR), 2003.

[13] In *NIPS 2003 workshop on feature extraction*, 2003.

[14] A. Popescul and L. H. Ungar. Structural logistic regression for link analysis. In *KDD Workshop on Multi-Relational Data Mining*, 2003.