Parallel Large Scale Feature Selection for Logistic Regression

Sameer Singh,
University of Massachusetts
Amherst MA 01003
sameer@cs.umass.edu

Jeremy Kubica, Scott Larsen,
Google Inc.
Pittsburgh PA 15213
{jkubica, esl}@google.com

Daria Sorokina
Carnegie Mellon University
Pittsburgh PA 15213
daria@cs.cmu.edu

Abstract

In this paper we examine the problem of efficient feature evaluation for logistic regression on very large data sets. We present a new forward feature selection heuristic that ranks features by their estimated effect on the resulting model's performance. An approximate optimization, based on backfitting, provides a fast and accurate estimate of each new feature's coefficient in the logistic regression model. Further, the algorithm is highly scalable by parallelizing simultaneously over both features and records, allowing us to quickly evaluate billions of potential features even for very large data sets.

1. Introduction

High-dimensional data sets with large number of features are used increasingly more often in real-world machine learning tasks. Text mining problems such as classification and spam detection rely on features that describe occurence of specific combinations of words and therefore the numbers of potential features can grow up to billions. Similarly, computational biology problems use inclusions of different amino acids sequences as features, and multiple ways to extract them result in large numbers of potential features. Another example of sparse high-dimensional data are link data sets, where each feature corresponds to an absence or presence of a link to some node in a large network, e.g. networks of co-authors or websites.

Recent comparison studies of machine learning algorithms in high-dimensional data revealed that the three top performing classes of algorithms for high-dimensional data sets are logistic regression, Random Forests and SVMs [6]. Although logistic regression can be inferior to non-linear algorithms, e.g. kernel SVMs, for low-dimensional data sets, it often performs equally well in high-dimensions, when the number of features

goes over 10000, because most data sets become linearly separable when the numbers of features become very large. Given the fact that logistic regression is often faster to train than more complex models like Random Forests and SVMs [16], in many situations it is the preferable method to deal with high dimensional data sets.

However, even with a scalable algorithm it can still be computationally infeasible to use the billions of features that could be potentially useful. The choice of features in high dimensions can have a significant effect on the performance of the learned model and the computational tractability of the learning algorithm. Many algorithm-independent high level feature selection techniques exist, however, in most cases the running time becomes an issue for large numbers of features.

In this paper we examine the problem of efficient feature evaluation for large scale logistic regression problems. In particular, we examine the setting of *forward* feature selection where on every step new features are added to an existing model.

We propose a framework for handling feature selection for logistic regression. First, we present an efficient scoring heuristic for new features that is based on evaluating the performance of an approximate model containing each new feature. Second, we present a highly parallelized algorithm for feature evaluation that is based on the map-reduce framework. Together these techniques provide a highly scalable feature evaluation algorithm, allowing us to efficiently scale to very large numbers of both records and features. Our framework makes use of a number of existing techniques such as forward selection, projection pursuit, Newton-Raphson approximation and map-reduce. To the best of our knowledge, combination of these techniques optimized specifically for logistic regression, is novel. The resulting algorithm is well suited for real-world problems with extremely large number of dimensions.

1.1. Logistic Regression

Logistic regression is a simple model for predicting a probability of event and is often used for binary classification. When the possible outcomes are coded as 0 and 1, we can train the logistic regression model that will predict the probability of the second event.

Assume that we have a data set $\{(\vec{x}_i, y_i)\}, 1 \leq i \leq N$, where $\vec{x}_i$ are the vectors of input feature values and $y_i \in \{0, 1\}$ are binary response values. Logistic regression represents log odds of the event as a linear model:

$$(1.1) \qquad log\left(\frac{p}{1-p}\right) = \vec{\beta} \cdot \vec{x}$$

Here p is the predicted probability $P(y = 1)$, and $\vec{\beta}$ is the vector of model parameters. It is equivalent to the following representation of p:

$$(1.2) \qquad p = P(y = 1) = f(\vec{x}, \vec{\beta}) = \frac{e^{\vec{\beta} \cdot \vec{x}}}{1 + e^{\vec{\beta} \cdot \vec{x}}}$$

Therefore the model is completely defined by the vector of parameters $\vec{\beta}$. These parameters are usually learned by maximizing the data's loglikelihood:

$$(1.3)$$
$$L(\mathbf{X}, \vec{\beta}) = \sum_{i=1}^{N} \left(y_i \ln f(\vec{x}_i, \vec{\beta}) + (1 - y_i) \ln(1 - f(\vec{x}_i, \vec{\beta})) \right)$$

Since there is no closed form solution to this maximization, the standard approach to solving it is to use an iterative algorithm such as Newton Raphson [13]. Although at the end we will converge to a very good approximation of the optimal solution, it can be a relatively time consuming process.

For the rest of the discussion we assume that the values of $\vec{x}$ are binary or continuous. For general categorical attributes, we use the standard technique of exploding the attributes into disjoint binary attributes. Thus an arity k feature becomes k disjoint binary features that form a logical grouping, called a *feature class*.

1.2. Forward Feature Selection

The goal of feature selection is to find a subset of features that produces the best[1] model $f(\vec{x})$ for

---

[1]In compliance with the logistic regression framework in this paper we focus primarily on defining "best" as the model with the maximum likelihood on the training or test set. However, the techniques described in the paper can easily be adapted to other scoring measures.

the data set $\{(\vec{x}, y)\}$. A naive approach for a set of D possible features would simply learn models for all 2^D possible combinations of features and directly evaluate their performance. However, the cost grows exponentially with the number of features and this method becomes completely unfeasible for even small feature sets.

Forward feature selection [22] is a heuristic that significantly reduces the number of models that we need to learn. We begin with an empty model and then on each iteration of the algorithm we choose a feature that gives the best performance when added to the current set of features. We refer to the process of selecting the best feature to add in an iteration of forward feature selection as *forward feature evaluation.* This means that on d-th iteration we need to build $D - d - 1$ models, where D is the original number of features, and the overall number of models to build and evaluate becomes quadratic. It is better than exponential, but the complexity is still very high when all $\vec{\beta}$ coefficients in every model are learned by a complex iterative method.

In addition to being a successful technique for feature selection as data preprocessing step, i.e., selecting the most useful subset from a large set of existing features, forward feature evaluation also has practical importance for the design of new features. Given the fact that often the number of possible new features is very large, it is important to have a fast way to estimate how useful they will be if introduced to the existing model. This is especially important in many real-world cases where adding new features to the data set may be expensive or labor intensive. In such cases forward feature evaluation can act as a quick filter on the features' effectiveness.

1.3. Paper Organization

The paper proceeds as follows: in §2 we present Single Feature Optimization (SFO), a new heuristic for feature evaluation that estimates the effect of new features on the model performance. We measure the effectiveness of new features by quickly learning an approximate model containing this feature and estimating the resulting change on a given performance metric. Further, in §3 we introduce an approach for parallelizing this and other feature-wise evaluation methods, allowing these methods to scale to large numbers of records and features. §4 shows an extensive evaluation of our technique on a number of artificial and real data sets. We also provide comparison with the results of other feature selection methods. We review related work in §5 and conclude in §6.

2. Evaluating the Features

Ideally we would like to evaluate each new feature by learning and evaluating a new model containing this new feature. However, as described above, this presents a computational challenge. For even moderate data set sizes and numbers of features, this may not be feasible.

In order to speed up the evaluation of a single feature we use the following heuristic: we retain coefficients from the current best model and optimize only the coefficient of the new feature β'_d. This produces an approximate model that can be evaluated. This way we would create $D - d - 1$ *approximate* models on each iteration of forward selection. After evaluating them to choose the best feature (or group of features) to add, we rerun the full logistic regression to produce an exact model that includes the newly added feature(s). We begin with this exact model for the next iteration of features selection, so the approximation error does not add up. As we rerun logistic regression only once on each iteration, we need to solve it now only D times — linear as opposed to quadratic.

2.1. Single Feature Optimization (SFO)

We can quickly learn an approximate model by limiting the optimization to only the new features. We hold the previous model parameters constant and perform a one dimensional optimization over the new coefficient. For each new feature x'_d, we compute an estimated coefficient β'_d by maximizing the loglikelihood with the new feature:

(2.4)
$$\operatorname*{argmax}_{\beta'_d} \sum_{i=1}^{N} \left(y_i \ln f_d(\vec{x}_i, \vec{\beta}) + (1 - y_i) \ln(1 - f_d(\vec{x}_i, \vec{\beta})) \right)$$

where $f_d(\vec{x}_i, \vec{\beta})$ denotes the logistic function over the original feature vector and the new feature:

(2.5)
$$f_d(\vec{x}_i, \vec{\beta}) = \frac{e^{\vec{\beta} \cdot \vec{x}_i + x'_{id} \beta'_d}}{1 + e^{\vec{\beta} \cdot \vec{x}_i + x'_{id} \beta'_d}}$$

This optimization is based on a single iteration of backfitting [5] or projection-pursuit regression [12]. We call this heuristic **S**ingle **F**eature **O**ptimization or SFO.

There are a variety of optimization approaches that we could use to solve Equation 2.4. We use Newton's method to solve:

(2.6)
$$\frac{\partial L}{\partial \beta'_d} = 0$$

We start at $\beta'_d = 0$ and iteratively update β'_d using the standard update:

(2.7)
$$\beta'_d = \beta'_d - \frac{\frac{\partial L}{\partial \beta'_d}}{\frac{\partial^2 L}{\partial \beta'^2_d}}$$

until convergence. In the case of optimizing the loglikelihood in (2.4) the derivatives simplify to:

(2.8)
$$\frac{\partial L}{\partial \beta'_d} = \sum_{i=1}^{N} x'_{id} (y_i - f_d(\vec{x}_i, \vec{\beta}))$$

(2.9)
$$\frac{\partial^2 L}{\partial \beta'^2_d} = - \sum_{i=1}^{N} x'^2_{id} f_d(\vec{x}_i, \vec{\beta})(1 - f_d(\vec{x}_i, \vec{\beta}))$$

It is important to note that this technique of optimizing along a single feature can also be used with other objective functions. As a very simple example we can add L_2-regularization to the above method by just including a penalty term in (1.3) and thus compute β'_d to maximize:

(2.10)
$$\sum_{i=1}^{N} \left(y_i \ln f_d(\vec{x}_i, \vec{\beta}) + (1 - y_i) \ln(1 - f_d(\vec{x}_i, \vec{\beta})) \right) - \lambda \beta'^2_d$$

and using the modified derivatives:

(2.11)
$$\frac{\partial L}{\partial \beta'_d} = \sum_{i=1}^{N} x'_{id} (y_i - f_d(\vec{x}_i, \vec{\beta})) - 2\lambda \beta'_d$$

(2.12)
$$\frac{\partial^2 L}{\partial \beta'^2_d} = - \sum_{i=1}^{N} x'^2_{id} f_d(\vec{x}_i, \vec{\beta})(1 - f_d(\vec{x}_i, \vec{\beta})) - 2\lambda$$

More generally, we could directly optimize a completely different metric, such as squared error. For consistency and simplicity, unless otherwise noted, we focus on optimizing unpenalized loglikelihood throughout this paper.

The obvious drawback of such single feature optimization is that we are not relearning the remaining coefficients. Therefore we may *underestimate* the performance of the new model on training set metrics. Despite this potential drawback, this limit optimization still can provide a strong signal. In particular, this approximation can equivalently be viewed as learning a single feature model to correct the previous logistic model.

2.2. Feature Class Optimization

Many real-world problems contain categorical attributes that can be exploded into a series of disjoint binary features. It is important to note that such explosions are particularly well suited for the single feature optimization described above.

Features from a single feature class are by definition disjoint. Since we are holding all of the other coefficients fixed, we can optimize each feature *independently* and later combine the resulting coefficients to form a complete model. Further, each of these optimizations only needs to run over those records containing the relevant feature. For an arity A categorical attribute that has been exploded into $\vec{x}' = \{\vec{x}'_1, \ldots, \vec{x}'_A\}$ we estimate $\vec{\beta}' = \{\vec{\beta}'_1, \ldots, \vec{\beta}'_A\}$ by maximizing:

(2.13)
$$\sum_{i:x'_{id}=1} \left(y_i \ln f_d(\vec{x}_i, \vec{\beta}) + (1 - y_i) \ln(1 - f_d(\vec{x}_i, \vec{\beta})) \right)$$

independently for each $0 \leq d < A$. Thus we can trivially break the problem of evaluating large arity categorical attributes into a series of smaller independent optimizations.

2.3. Scoring the Features

Once we have the approximate model containing the features in the new feature class $\vec{\beta}^+ = \{\vec{\beta}, \vec{\beta}'_1, \ldots, \vec{\beta}'_A\}$, it is trivial to use it to compute standard performance metrics such as log-likelihood, AUC, or prediction error. Thus we can score the new feature class by the directly evaluating the approximate model.

2.4. Faster Optimization with Histograms

As the number of training records increases even the the one dimensional optimization described in § 2.1 can require a non-trivial amount of computation. During each step of the Newton's method, we are required to run through all records containing an attribute. For binary or exploded categorical attributes can further reduce this cost by using an approximate optimization based on histograms.

Instead of performing the optimization directly from the records, we can form a histogram over *predicted probabilities* and use this as the basis of our approximation. We store two histograms with an equal number of B bins for each attribute, binned on the predicted probabilities of the *base model*. Each bin b tracks the number of records falling into a given range of predicted probabilities N_b and the number of those records that also have positive outcomes N_b^+. Along with the

counts, it is trivial to store the original activation value $a = \beta' \cdot \vec{x}$ that would have produced the midpoint probability of that bin p_b:

(2.14)
$$a_b = \log \left(\frac{p_b}{1 - p_b} \right)$$

From the activation we can compute the modified predicted probability accounting for the new attribute:

(2.15)
$$p'_b = \frac{e^{a_b + \beta'_d}}{1 + e^{a_b + \beta'_d}}$$

Thus we only need to perform the optimization over the bins in the histogram, using the modified derivative computations:

(2.16)
$$\frac{\partial L}{\partial \beta'_d} = \sum_{b=1}^{B} N_b^+ - p'_b \cdot N_b$$

(2.17)
$$\frac{\partial L}{\partial \beta'^2_d} = -\sum_{b=1}^{B} N_b \cdot p'_b \cdot (1 - p'_b)$$

This approximation brings the computational cost of each iteration of Newton's method from $O(N)$ down to $O(B)$. This can be a significant savings for problems with a large number of records where $N >> B$. The space and time costs of the histograms can be kept low by using a sparse representation and an efficient binning scheme.

The use of histograms introduces an explicit tradeoff between model accuracy and computational cost. Increasing the number of bins provides a more accurate representation of the probability distribution, but also increases the cost of computing the Newton's step over these bins. The appropriate tradeoff and value of B depends on the problem constraints, number records, and distributions of the values of β'_d. A full analysis of the tradeoffs is outside the scope of this paper. However, it is easy to empirically examine the tradeoffs on a particular data set by varying B and computing the change in both the estimated coefficients and the algorithm's running time.

3. Parallelization

In order to scale up to very large data sets, in terms of both records and potential features, we can parallelize the feature evaluation algorithm. In particular, we developed the SFO evaluation algorithm in the context of the map-reduce framework [8]. As described below, the map-reduce framework consists of two distinct phases, mapping and reducing, which parallelize
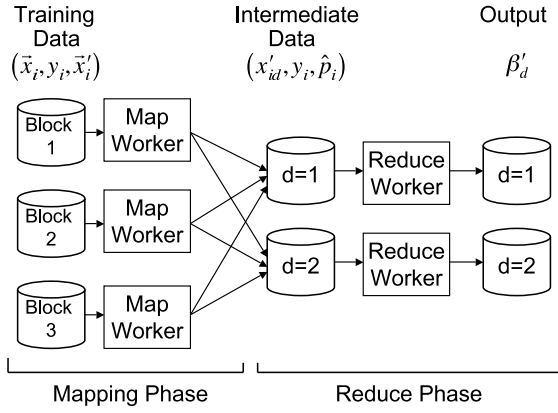
Figure 1. Conceptual data flow of the feature evaluation map-reduce with 3 input data blocks and 2 features. In the mapping stage separate processors operate on blocks of training data $(\vec{x}_i, y_i, \vec{x}'_i)$ to produce intermediate data records (y_i, p_i) for each new feature in the record $\vec{x}'_i$. In the reduce phase separate processors operate on each of the intermediate data sets, computing estimated coefficients for the new features β'_d.

MapFunction$(\{\mathbf{X}, \vec{y}\}, \vec{\beta})$
Input: A data block $\{\mathbf{X}, \vec{y}\}$ and model $\vec{\beta}$.
Output: Intermediate data sets $T_d \forall d$.
1 FOR each $\{\vec{x}_i, y_i, \vec{x}'_i\}$ in $\{\mathbf{X}, \vec{y}\}$:
2      $p_i = f(\vec{x}_i, \vec{\beta})$
3   FOR each $x'_{id} \in \vec{x}'_i$:
4   Store (y_i, p_i) in the intermediate data T_d
5          $T_d = T_d \cup (x'_{id}, y_i, p_i)$

ReduceFunction(T_d)
Input: An intermediate data set T_d.
Output: Estimated coefficient β'_d.
1 $\beta'_d = 0$
2 Until convergence of β'_d:
3      $\frac{\partial L}{\partial \beta'_d} = \frac{\partial^2 L}{\partial \beta'^2_d} = 0$
4   FOR each $(x'_{id}, y_i, p_i) \in T_d$:
5          $a_i = \log\left(\frac{p_i}{1-p_i}\right)$
6          $p'_i = \frac{e^{a_i + \beta'_d}}{1 + e^{a_i + \beta'_d}}$
7          $\frac{\partial L}{\partial \beta'_d} = \frac{\partial L}{\partial \beta'_d} + (y_i - p'_i)x'_{id}$
8          $\frac{\partial^2 L}{\partial \beta'^2_d} = \frac{\partial^2 L}{\partial \beta'^2_d} - p'_i(1 - p'_i)x'^2_{id}$
9      $\beta'_d = \beta'_d - \frac{\partial L}{\partial \beta'_d} / \frac{\partial^2 L}{\partial \beta'^2_d}$

Figure 2. The map-reduce SFO algorithm.

the computation over the training records and potential features respectively.

The SFO map-reduce algorithm, illustrated in Figure 1 and code given in Figure 2, consists of three steps:

1. **Mapping Phase** (*parallel over records*): Iterate over the training records $(\vec{x}_i, y_i, \vec{x}'_i)$, computing which new features occur in $\vec{x}'_i$ and the predicted probability of the current model $p_i = f(\vec{x}_i, \vec{\beta})$. Each new feature present in the input record produces an intermediate data record (x'_{id}, y_i, p_i). These intermediate records are aggregated in disjoint sets for each new feature.

2. **Reduce Phase** (*parallel over features*): For each feature being evaluated β'_d use the corresponding outputs of the mapping phase, (x'_{id}, y_i, p_i), to compute an estimated coefficient β'_d as in § 2. We can also aggregate estimated changes in training set loglikelihood by using the estimated coefficients.

3. **Post-processing**: Aggregate the coefficients for all features in the same feature class.

Since the features are always treated independently up to the post-processing phase, we can also use this algorithm to evaluate many different non-disjoint feature
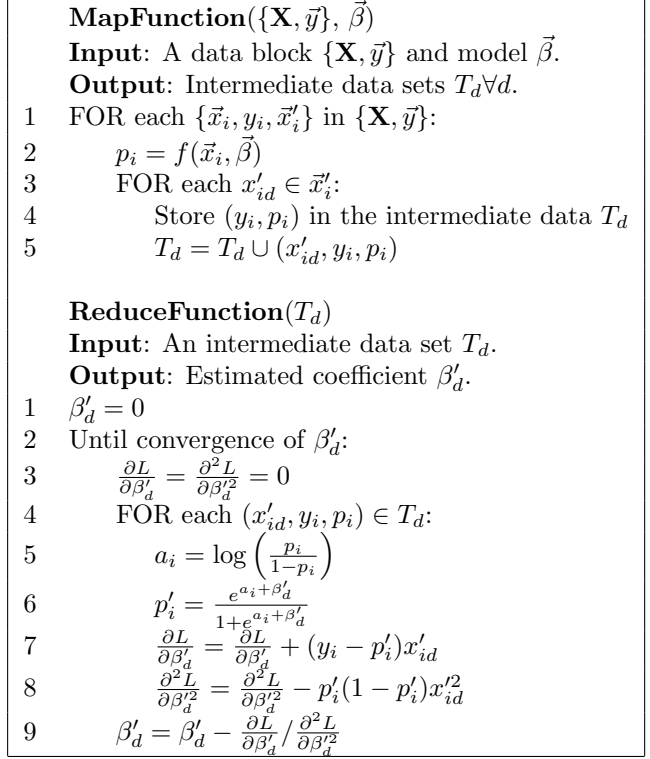
classes in a single run. This allows us to trivially explore many potential feature classes in parallel.

The running time of the mapping phase is approximately $O(\frac{N \cdot D_{\max}}{C})$ where N is the number of records, $D_{\max}$ is the maximum number of features active in any record, and C is the number of machines used. Similarly, the running time of the reduce phase is $O(\frac{N_{\max} \cdot D}{C})$ where $N_{\max}$ is the maximum number of records containing a single new feature. In real world systems there is also a (non-trivial) constant per-machine start up cost that bounds the contribution of adding more machines.

We can also use the same framework to compute test set metrics with a second map-reduce-based algorithm. In this case the algorithm knows the estimated coefficients $\vec{\beta}'$ and the phases become:

1. **Mapping Phase** (*parallel over records*): Iterate over the test records $(\vec{x}_i, y_i, \vec{x}'_i)$ and for each new feature $\vec{x}'_{id}$: compute the predicted probabilities under the old model $p_i = f(\vec{x}_i, \vec{\beta})$ and the new model $p'_{id} = f_d(\vec{x}_i, \vec{\beta}^+)$.

2. **Reduce Phase** (*parallel over features*): For each feature being evaluated β'_d use the model with and without the new coefficient to compute the

difference in score.

3. **Post-processing**: Aggregate the score changes for all features in the same feature class.

Finally, it is important to note that this framework can also be applied to other feature evaluation methodologies. For example, we applied the gradient based approach of Perkins et al. [20]. In this case the mapping phase iterates over the records computing the gradient for each active feature. The reduce phase aggregates these gradients to produce the overall gradient for each feature.

4. Experimental Results

The feature evaluation problem can be rephrased as follows. Let the base data set be the data set containing some of the feature classes (F_B) from the complete set of features (F). Additional features ($F_E = F - F_B$) have to be evaluated, and the most useful feature from F_E has to be selected for inclusion into the model. We conducted a series of experiments to test SFO's effectiveness in determining which feature class to add to a current model. Because we are specifically interested in efficiently approximating the performance of a single step of forward feature selection, our experiments focus on the task of adding one new feature.

For a baseline comparison we used the standard forward selection wrapper method: retrain the entire model with the new feature. This allows us to compare the SFO approximation with the full forward selection wrapper method. We used a publicly available IRLS logistic regression package[2] to learn the base model and the full retrained models [16]. The classifier is retrained on data with features $F_B + f_i$ (where f_i is a single feature class from F_E). The feature classes f_i are then ranked according to performance of the resulting classifier. For some experiments, we also compare our evaluation with the gradient method proposed by Perkins *et. al.*, which ranks the feature classes according to their gradient on the training set [20].

4.1. Simulated Data

We first tested the different feature selection algorithms on simulated data from random logistic models. The goal of the experiments is to find the single new relevant feature within a set of irrelevant features. Thus the models were created using a number of relevant base features, F_B, with $|\beta| \in [0.2, 1.5]$, one relevant experimental feature with $|\beta| \in [0.2, 1.5]$, and a number of

$|F_B|$	$|F_I|$	Training Set			Test Set	
		IRLS	SFO	GD	IRLS	SFO
50	5	17	17	17	15	14
50	10	16	15	13	12	9
50	50	16	11	11	12	10
50	100	18	16	14	17	14
50	200	18	18	11	9	10
50	500	13	13	9	9	5
1	50	18	19	13	15	14
5	50	19	17	15	13	10
10	50	16	11	11	12	10
20	50	18	17	14	14	12
50	50	14	12	12	11	6
100	50	12	10	10	7	6

Table 1. Performance of the feature selection algorithms as the number of base F_B and irrelevant F_I features increase. Performance is the number of runs, out of 20 total, in which the true relevant feature is successfully found.

irrelevant experimental features, F_I, with $|\beta| \leq 0.02$. All coefficients had a 50% probability of being negative and all features had independent, randomly chosen occurrence probabilities in the data between 0.05 and 0.40. The algorithm's performance was measured by the number of runs in which the algorithm correctly ranked the relevant feature as the best one to add.

We examined the effect of increasing both the set of irrelevant attributes F_I and the set of base features F_B. Increasing either of these dimensions should make the problem more difficult, allowing us to test the algorithm's robustness. In particular, as the size of the base model increases the marginal effect of a new feature should *decrease*, making the new feature harder to find. For both of these experiments we used 2000 data records, a random 10% test set, and no regularization. The results are shown in Table 1.

Table 1 shows several interesting trends. First, as expected, resolving the full logistic regression with IRLS gives the best performance. However, the performance on the SFO heuristic is often equal to or practically close to that of IRLS. Second, the performance of the feature evaluation algorithms is robust to the number of irrelevant features, but decreases with the size of the base model. Third, the test set metrics significantly underperform the training set metrics. This last point is unsurprising given the very simple nature of the problem and the small test set sizes. Additional experiments showed that with a 50/50 split the two metrics were significantly closer.

4.2. UCI data sets

We also examined the performance of the algorithms on two data sets from the UCI repository: the mushroom and internet ads data sets [3].

Mushroom. The mushroom data set contains 8124 instances, 22 nominal features, and binary class labels (poisonous and edible). One of the features had a number of missing values and was filtered out. We used a 10% test set in all of the experiments.

This is a good data set for testing feature evaluation methods since some fairly simple features can predict the class with high accuracy. For example, the rule:

```
odor = NOT (almond.OR.anise.OR.none)
```

gives an accuracy of 98.52%. Since our features are evaluated independent of each other, such rules are easily captured by the method.

Our first experiment consisted of simulating the first two rounds of feature selection. First we started with just the bias term ($F_B = \{$ bias $\}$) and evaluated all the features using our method, i.e. $F_E = F$. Table 2 shows the top 5 features according to IRLS, with their test set log-likelihood estimates, and the respective values and rankings when estimated using SFO. We observe that, in compliance with the data set description, the most informative feature was determined by both methods to be `odor`. Table 2 also shows the top 5 features from the second round when we start with $F_B = \{$ bias, odor $\}$. It should be noted that in both rounds the features are ranked almost identically by the IRLS and SFO. Further, both methods have an identical choice for the best feature to add.

We also tested the case where we exploded the feature space by including all 232 possible conjunctions of feature classes: $(f_i, f_j) \ \forall i \neq j$. Again, the base set of features only included the bias term. In this case we would expect conjunctions containing `odor` to provide several very close "almost equally good." options. As a result we would expect the SFO approximation to be a larger factor and lead SFO to choose a different feature from IRLS. The results of this run matched expectations with SFO and IRLS producing different orderings among the top features.

Internet Ads. Internet Ads is a larger set containing 1558 features (3 continuous, and rest binary) and 3279 instances. We divided each continuous feature into 10 binary features using linear binning. The first 500 feature classes (i.e. 527 binary features) were used as the base data set, and the rest of the 1059 features were evaluated. As before, we examine the features ranked by test set log-likelihood of SFO and the gradient method, and compare these selections by retraining IRLS.

F_B	Feature Class	IRLS -LL	SFO -LL	SFO Rank	GD Rank
bias	odor	0.111	0.076	1	2
	spore-print-color	0.558	0.543	2	1
	gill-color	0.623	0.604	3	9
	stalk-surface-above	0.696	0.692	5	3
	ring-type	0.711	0.687	4	8
bias, odor	spore-print-color	0.074	0.069	1	5
	stalk-surface-above	0.098	0.090	3	3
	population	0.099	0.092	5	6
	gill-color	0.099	0.091	4	7
	stalk-color-below	0.100	0.086	2	4

Table 2. The negative test set log-likelihood for the top features in the Mushroom data set as selected by IRLS, the corresponding SFO scores, and rankings from SFO and the gradient method.

Figure 3 shows the coverage for the complete ranking. Formally, we define the coverage at level x as the percentage of the top x features selected by an algorithm that are also selected by IRLS. Thus we use the IRLS wrapper method as a ground truth comparison.

As shown, the SFO method is much better than the gradient method throughout the range of the list. It is especially important to note that SFO correctly chooses the top 4 most important features from a choice of more than a thousand. On the other hand, the gradient method fails in ranking the features correctly, producing a completely different selection from IRLS for the first 79 features.

Table 3 shows the ranking of the top 3 features for each algorithm on several metrics. Note that SFO produces rankings that are very similar to those of full forward feature selection on all of the metrics. In the one case where the the two algorithms chose different "best features" the corresponding AUC scores were close enough that SFO's approximation *did* matter. In addition the numeric scores, which are not shown for clarity, are also very close indicating that we are getting good approximations.

4.3. RCV1-v2

To demonstrate a more realistic data size for the distributed SFO heuristic, we applied the algorithm to the RCV1-v2 data [18]. This data consists of stemmed tokens from text articles on a range of topics. We used the predefined training set, which contains 23,149 records and 47,152 features. As in [4], we used the "economics" (`ECAT`) category as the positive cases,

Rank	Training LL		Training AUC		Test LL		Test AUC		GD
	IRLS	SFO	IRLS	SFO	IRLS	SFO	IRLS	SFO	
1	1243	1243	1483	1243	1243	1243	1243	1243	1385
2	1399	1399	1243	1483	1399	1399	968	968	*
3	1483	1344	968	1399	968	968	1399	1399	*
Overlap	2		2		3		3		0

Table 3. The top 3 features using the different algorithms and evaluation metrics. The overlap indicates how many of the top three features are in agreement with IRLS. *Gradient descent had a 4-way tie among 1040, 1146, 1309, and 1391 for second place.
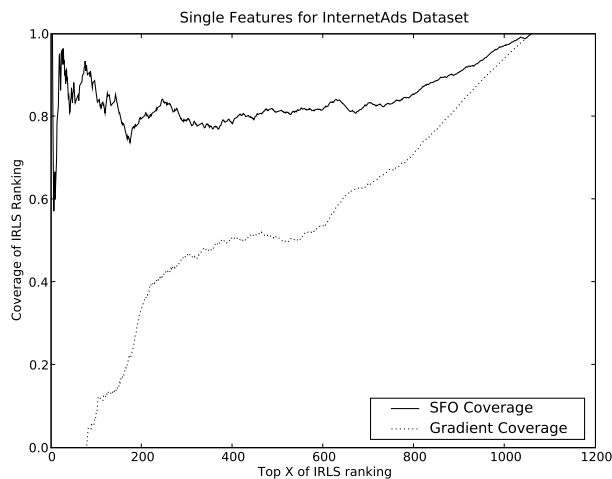
Figure 3. Coverage of the IRLS ranking by SFO and the Gradient method for the Internet Ads data. The features were ranked by test set log-likelihood.

giving us 3,449 positive training instances.

As an initial experiment, we examined the initial features selected by SFO. Our initial base model was just the bias term and after each round we constructed a new base models with the features selected so far. Table 4 shows the top 5 results of SFO after each of the first 5 rounds under different levels of L_2-regularization.

Table 4 shows several interesting and expected trends. First, the ordering of the attributes changes as we add features to the model. The use of a score such as loglikelihood means that we automatically down-rank features that no longer add as much expected benefit to the model, such as those correlated to features already in the base model. This effect is evident in the addition of "shar", which is in the first 5 attributes added despite not being ranked highly during the first round.

Second, we can clearly see the effect of regularization on the feature selection. For example, "shar" is moved from the fourth feature added up to the third when $\lambda = 100$. At the same time the estimated benefit of the features decrease as the regularization dampens the 1-dimensional optimization.

Finally, the improved performance of the underlying base model can be seen in the decreasing estimate benefit of new features. As the base model improves during each round each new feature is being targeted at just the residuals of the current model. A similar effect can be seen in the increasing stability of base model's parameters, such as the bias term.

We also compared the first 5 features that we add to our model with the features that Balakrishnan and Madigan found to have largest coefficients in their fully trained and regularized model [4]. Although this is not a fair comparison, because we are solving different problems, it still provides an interesting check as to our ability to find features that are useful in the final model. Three of the first five features added by SFO (econom, shar, and inflat), were listed in their top 10.

Finally, we looked that how the top ranked features change as we are allowed to use more data. The RCV1-v2 data set is pre-divided into single training set and four test sets. We ran the first two rounds of feature selection on an increasing amount of training data by using the records in the training set and 0 to 4 of the tests sets. This effectively performed feature selection using 3%, 28%, 52%, 77%, and 100% of the available data. The features estimated without regularization and were ranked by changes in training set loglikelihood.

The results are shown in Table 5. As expected, different features are ranked higher as we increased the data size from 3% to 28%. In particular, `municip` was chosen as the second feature when using 3%, but `shar` was chosen in all other cases. However, after increasing to 28% of the data the rankings are roughly stable. It is also interesting to note that with the increased

λ	Round 1		Round 2		Round 3		Round 4		Round 5	
	econom	283.7	municip	204.3	deficit	110.2	*shar*	106.7	inflat	79.5
	deficit	213.7	*shar*	139.3	*shar*	106.8	statist	82.1	wag	77.1
0.0	inflat	190.1	coupon	131.3	statist	90.2	wag	79.7	statist	76.5
	gdp	182.9	obligat	110.3	inflat	87.2	*profit*	79.1	moody	68.6
	municip	176.3	*profit*	106.1	gdp	86.6	inflat	74.7	*digest*	66.0
	econom	283.5	municip	201.4	deficit	108.7	*shar*	106.1	inflat	78.5
	deficit	211.6	moody	189.8	*shar*	106.3	statist	80.3	statist	74.9
10.0	inflat	188.4	*shar*	138.7	statist	88.3	*profit*	77.8	wag	74.6
	gdp	177.7	coupon	129.2	inflat	86.2	wag	77.2	moody	66.4
	budget	165.0	obligat	105.6	gdp	83.4	inflat	72.8	coupon	56.6
	econom	269.4	municip	133.5	*shar*	91.2	deficit	76.6	inflat	54.3
	deficit	153.7	*shar*	120.4	deficit	76.0	inflat	64.9	presal	50.4
100.0	inflat	139.3	moody	116.1	*clos*	62.4	budget	58.4	*clos*	47.0
	budget	132.9	coupon	80.7	inflat	61.8	statist	50.6	statist	45.1
	municip	118.4	*profit*	80.1	*profit*	60.8	balanc	49.9	pct	44.2

Table 4. The top 5 features and their estimated improvement in training set loglikelihood for the first 5 rounds of feature selection with two different levels of L_2-regularization. The features in italics have negative estimated coefficients.

amount of data, SFO's two initial feature selections match the largest coefficients found by Balakrishnan and Madigan's RMMP algorithm on their training set + 2 test set case [4].

This last experiment also demonstrates the potential scalability of this approach over a traditional wrapper algorithm. Fully relearning the models, would require solving 288,062 logistic regressions over 804,414 records. Instead we are able to parallelize the approach and only touch each record a single time.

4.4. Timing Results

To test the effectiveness of using the map-reduce framework, we examined the wall clock running time of the algorithm as we varied the number of machines. We computed the time required *relative* to the single machine implementation. We used two data sets, one with 1,000,000 records and 50,000 features, and the other with 10,000,000 records and 100,000 features. The true β coefficients were randomly generated from the range $[-0.5, 0.5]$. Each record contains exactly 20 active features selected randomly without replacement. The base model has a prior probability of 0.5.

Figure 4 shows a clear benefit as we increase the number of machines. The deviation from ideal when using higher number of machines in the Speed Up plots 4(b) and 4(d) occurs since the benefit of adding machines decreases as the constant startup costs begin to become an increasing factor. Despite the decreasing re-

turns with the number of machines, we can expect this parallelization to provide significant wall clock savings for large data sets or feature sets. This can be observed by noticing that the deviation from ideal is less for the larger dataset (for the same number of machines). Further, we expect the marginal benefit of adding machines to *increase* with the computational cost of computing the features. This type of parallelization becomes more important when we consider non-trivial features, such as those requiring string parsing.

5 Related Work

Our work is based on *forward selection* framework. Forward selection was introduced by Whitney [22]. Such wrapper algorithms are expected to perform well, but often at the cost of high computational complexity [14]. An opposite approach, backward elimination, also suffers from similiar problems: Abe [1] discusses the benefits and computational costs of a modified backward feature selection algorithm and ultimately suggests batching the features to improve the efficiency.

Della Pietra et al. [9] describe a feature selection method for random fields that holds the features in the current model fixed and selects the new feature by minimizing the KL-divergence of the model with the empirical data. McCallum [19] introduces a similar method for conditional random fields, but his algorithm chooses the feature that maximizes the new model's loglikeli-

	0% Test Set	25% Test Set	50% Test Set	75% Test Set	100% Test Set
N	23,149	222,477	421,816	621,392	804,414
D	47,152	146,582	204,052	249,587	288,062
	econom	econom	econom	econom	econom
Round 1	deficit	deficit	deficit	deficit	budget
Top 5	inflat	budget	budget	budget	deficit
Features	gdp	gdp	gdp	gdp	gdp
	municip	monet	inflat	monet	monet
	municip	shar	shar	shar	shar
Round 2	shar	municip	municip	municip	municip
Top 5	coupon	moody	moody	moody	moody
Features	obligat	deficit	deficit	budget	budget
	profit	budget	budget	deficit	coupon

Table 5. Top 5 features for the first two rounds of feature selection with different training sizes.

hood instead. Both of these techniques use Newton's method to solve the resulting one-dimensional optimization problems.

Several other works have also used feature-wise evaluation functions that incorporate predictions of the existing model. Perkins et al. [20] employ feature's gradient as part of their Grafting algorithm. Fleuret [11] developed a filter method based on conditional mutual information. It chooses features that maximize the minimum mutual information with the response variable conditioned on each of the features already in the model.

Eyheramendy and Madigan [10] invented the Posterior Inclusion Probability approach. It is based on knowledge of the type of model that will be used (e.g. Poisson, Bernoulli, etc.), but only in closed forms.

Some of the feature selection techniques described above bear similarity to our algorithm, however, our work goes beyond these approaches. First, we use the resulting approximate model to rank the features on potential range of metrics, such as AUC or test set performance. Second, we present a highly parallel algorithm that allows us to easily scale to very large data sets.

In addition to the methods mentioned above, there exists a wide range of alternate feature selection algorithms. Those include *filter methods*, such as RE-LIEF [15] and FOCUS [2], where feature evaluation is independent of the underlying learning algorithm. Another approach to feature selection is to automatically select features through regularization or feature priors. LASSO [21] is a well known algorithm of this type based on L_1 penalty. More recently Krishnapuram et al. [17] proposed Bayesian methods for joint learning and feature selection. For a good survey of other classes of feature selection techniques we refer the interested reader to [7].

## 6. Conclusions and Future Directions

In this paper we presented a new heuristic and a highly parallel algorithm for forward feature evaluation. The SFO heuristic provides an approximation of the effect of adding a new feature to the model. Empirically, we showed that this heuristic results in a good performance and is comparable to techniques that relearn the whole model. Unlike other feature-wise approaches, the approximate model can also be used to evaluate the feature's performance on a range of other metrics and on validation set performance. Further, the coefficients estimated by the SFO heuristic can provide useful starting points to relearn the model and can provide insights into the structure of the problem.

We described a highly parallel algorithm, based on the map-reduce framework, for performing feature evaluation. This approach makes feature evaluation tractable on massive data sets. Further, it can trivially be applied to the SFO heuristic as well as other known heuristics. We showed that using approximate log-likelihood computations based on histogram data can further reduce the computational cost.

In the future, we would like to extend this work to more complex feature evaluation tasks. For example, limited correlation data could be maintained to facilitate evaluating pairs of feature classes. Alternatively similar techniques may be applied to backward feature selection, allowing us to quickly perform informed random walks in model space. In addition, we would like to apply SFO to directly optimizing other metrics.
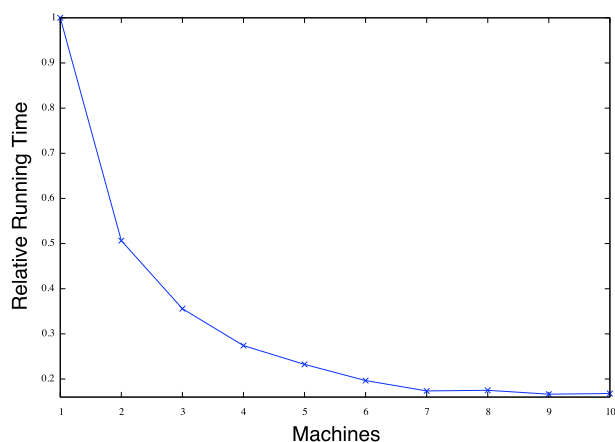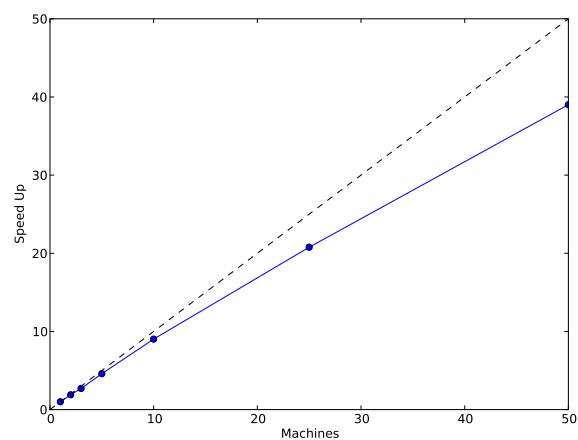
(a) Relative Running Time (1,000,000 records / 50,000 features)
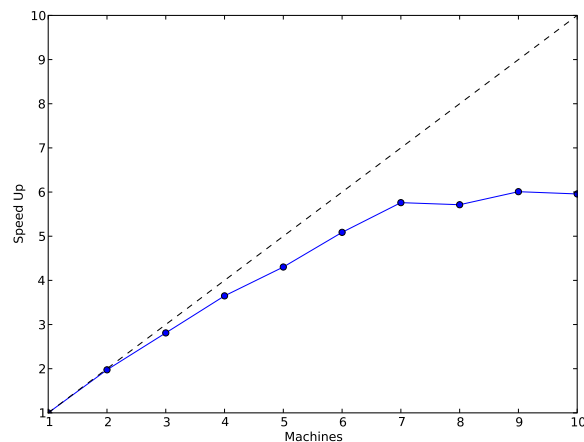
(b) Speedup (1,000,000 records / 50,000 features)
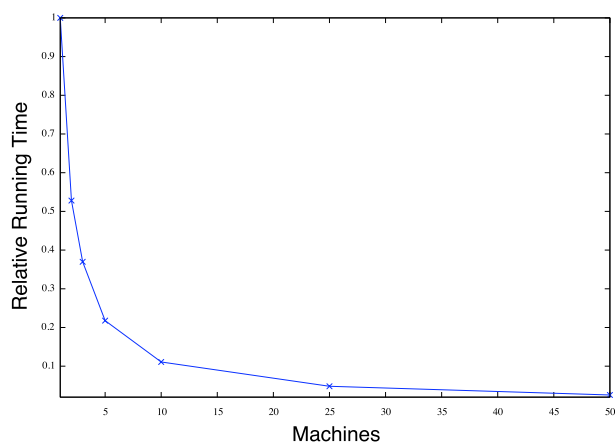
(c) Relative Running Time (10,000,000 recs / 100,000 feats)

(d) Speedup (10,000,000 records / 100,000 features)

Figure 4. Relative Time and Speed plots of the distributed SFO evaluator versus the number of machines used for 1,000,000 records / 50,000 features (a and b) and 10,000,000 records / 100,000 features (c and d). The Speed Up plots are inverse of the relative time plots, where the dashed line represents ideal behavior.

References

[1] S. Abe. Modified backward feature selection by cross validation. In *13th European Symposium On Artificial Neural Networks*, 2005.

[2] H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 547–552. AAAI Press, 1991.

[3] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[4] S. Balakrishnan and D. Madigan. Algorithms for sparse linear classifiers in the massive data setting. *Journal of Machine Learning Research*, 2008. to appear.

[5] A. Buja, T. J. Hastie, and R. J. Tibshirani. Linear smoothers and additive models (with discussion). *Annals of Statistics*, 17:453–555, 1989.

[6] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning, (ICML 2008)*, 2008.

[7] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis - An International Journal*, 1(3), 1997.

[8] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, 2004.

[9] S. Della Pietra, V. Della Pietra, and J. Lafferty. In-

ducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

[10] S. Eyheramendy and D. Madigan. A novel feature selection score for text categorization. In *Proceedings of the International Workshop on Feature Selection for Data Mining: Interfacing Machine Learning and Statistics*, April 2005.

[11] F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5:1531–1555, 2004.

[12] J. H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823, December 1981.

[13] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, 2001.

[14] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning, (ICML 1994)*, pages 121–129. Morgan Kauffmann, 1994.

[15] K. Kira and L. A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 129–134. MIT Press, 1992.

[16] P. Komarek and A. Moore. Making logistic regression a core data mining tool with tr-irls. In *Proceedings of the 5th International Conference on Data Mining Machine Learning*, page 4, 2005.

[17] B. Krishnapuram, L. Carin, and A. J. Hartemink. Joint classifier and feature optimization for comprehensive cancer diagnosis using gene expression data. *Journal of Computational Biology*, 11(2-3):227–242, March 2004.

[18] D. D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[19] A. McCallum. Efficiently inducing features of conditional random fields. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

[20] S. Perkins and J. Theiler. Online feature selection using grafting. In T. Fawcett and N. Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 592–599. AAAI Press, 2003.

[21] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58(1):267–288, 1996.

[22] A. W. Whitney. A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, 20(9):1100–1103, 1971.