



The University of Sydney

A Hill-climbing Landmarker Generation Algorithm Based on Efficiency and Correlativity Criteria

Technical Report Number 560

November 2004

Daren Ler, Irena Koprinska and Sanjay Chawla

ISBN 1 86487 697 2

**School of Information Technologies
University of Sydney NSW 2006**

A Hill-climbing Landmarker Generation Algorithm Based on Efficiency and Correlativity Criteria

Daren Ler, Irena Koprinska, and Sanjay Chawla

University of Sydney

School of Information Technologies, Madsen Building F09, University of Sydney, NSW 2006, Australia
{ler, irena, chawla}@it.usyd.edu.au

Abstract

For a given classification task, there are typically several learning algorithms available. The question then arises: which is the most appropriate algorithm to apply. We recently proposed a new algorithm for making such a selection based on landmarking - a recent meta-learning strategy that utilises meta-features that are themselves efficient learning algorithms. This algorithm, which creates a set of landmarks that each utilise subsets of the algorithms being landmarked, was shown to be able to estimate accuracy well, even when employing a small fraction of the given algorithms. However, that version of the algorithm had an exponential training time computational complexity.

In this paper, we propose a hill-climbing version of the landmarker generation algorithm, which requires only polynomial training time complexity. Our experiments show that the landmarks formed, have similar results to the more complex version of the algorithm.

1. Introduction

The choice of machine learning algorithm can be a vital factor in determining the success or failure of a learning solution. Theoretical [20], as well as empirical results [13] have shown that no single learning algorithm is generically superior, meaning that we cannot rely on one particular algorithm to solve all our learning problems. Traditionally, selection is done based on some form of brute-force holdout testing (e.g. cross-validation and bootstrapping) [16]. However, with the growing plethora of machine learning algorithms, and because several different representations are usually reasonable for a given problem, such evaluation is often computationally unviable. As an alternative, some meta-learning [7, 17] techniques utilise past experience or meta-knowledge [7] in order to learn when to use which learning algorithm.

As with standard machine learning, the success of meta-learning is greatly dependent upon the quality of the features chosen. And while various strategies for defining such meta-features have been proposed [1, 3, 4, 8, 9, 13], there has been no consensus as to what such features might be.

Landmarking [6, 15] is a new and promising approach that characterises datasets by directly measuring the performance of simple and fast learning algorithms called landmarks. However, the selection of landmarks is typically done in an ad hoc fashion, with the landmarks generated focused on characterising an arbitrary set of algorithms. In previous work [11, 12], we had reinterpreted the role of landmarks, defining each as a set of learning algorithms that characterises the domain of expertise of one specific learning algorithm. As criteria, we specified that these landmarks should each be both efficient and correlated (as compared with the landmarked algorithm). However, the landmarker generation algorithm proposed (based on the all-subsets regression technique [14]) has a training time computational complexity that is exponential, making it unwieldy.

In this paper, we propose an alternate hill-climbing landmarker generation algorithm based on the forward selection [14] method for variable selection in regression. We show that this version of the landmarker generation algorithm performs almost as well as the all-subsets version, while requiring polynomial (as opposed to exponential) training time complexity.

In the next section, we describe landmarks and the criteria we have proposed to select them. This is followed by a description of the proposed hill-climbing landmarker generation algorithm in Section 3. Section 4 describes the experiments and discusses the results.

2. Landmarking and Landmarkers

In the literature [6, 15], a landmarker is typically associated with a single algorithm with low computational complexity. The general idea is that the performance of a learning algorithm on a dataset reveals some characteristics of that dataset. Correspondingly, given the pattern of performance of a set of algorithms over several datasets, we may be able to learn how various algorithms within that set will perform given the performance of the other algorithms in that set. However, in previous Landmarking work [15], despite the presence of two landmarker criteria (i.e. efficiency and bias diversity), no actual mechanism for

generating appropriate landmarks were defined, and the choice of landmarks was made in an ad hoc fashion.

Subsequently, in [11, 12], we redefined landmarking and landmarks. Given that: (i) a landmarking algorithm is an algorithm whose performance is utilised as a dataset characteristic (i.e. essentially the old definition of a landmarker), and (ii) a landmarker is a function over the performance of a set of landmarking algorithms, which resembles the performance (e.g. accuracy) of one specific learning algorithm. Then landmarking is thus the process of generating a set of algorithm estimators, called landmarks, such that each landmarker resembles the performance of one within a set of candidate algorithms being landmarked.

Consequently, we also suggest new landmarker criteria. Previously [15], it was suggested that each landmarking algorithm be (i) efficient (with maximum complexity $O(n^2)$), as well as (ii) bias diverse with respect to the other landmarking algorithms. However, as landmarker (and not landmarking algorithm) criteria, there is no guarantee that the landmarks selected via these criteria will be able to characterise the performance of one, and even less so, a set of candidate algorithms [11]. Instead, in [11, 12], we proposed the following landmarking criteria:

- **Correlativity.** Each landmarker should as closely as possible resemble the candidate algorithm being landmarked; fluctuations in the performance of the candidate algorithm should correlate to similar fluctuations in the landmarker.
- **Efficiency.** Given the set of candidate algorithms being landmarked, the computational cost of running the corresponding set of landmarks (on any given dataset) should be less (and preferably significantly less) than the computational cost of running those candidate algorithms (on the same dataset).

2.1 Selecting Landmarkers

Intuitively, two algorithms whose performance patterns are similar (i.e. given many datasets, their performance on each is similar) will be closer to each other in a performance space. Conceptually, the distance between two algorithms a and l can be regarded as $\|a - l\|$. Also, we may express $\|a - l\|^2$ as $\|a\|^2 + \|l\|^2 - 2 a.l$. Thus, if a is close to l , this implies that $\|a - l\|^2$ is small, and thus that $a.l$ is relatively large. This pertains to the correlativity criterion we use to check if a landmarker (e.g. l) is representative of some candidate algorithm (e.g. a). However, in order to operationalise $a.l$, we must move into the space of datasets, as depicted in Figure 3. Thus, we may measure correlativity based on r [18] as:

$$r = a.l = \cos \angle(a, l) = \frac{a.l}{\|a\| \|l\|} = \text{cov}(a, l) / \sqrt{\text{var}(a) \cdot \text{var}(l)}$$

Several other heuristics similarly apply, including r^2 (i.e. the squared Pearson's correlation coefficient), Mallows's C_p criterion, Akaike and Bayesian Information Criteria (i.e. AIC and BIC respectively) [14]. In terms of efficiency,

simply utilising the (computational) time taken to run a given landmarker would suffice. However, aggregating the two into a single heuristic is more difficult; several methods to do this have been proposed in [10].

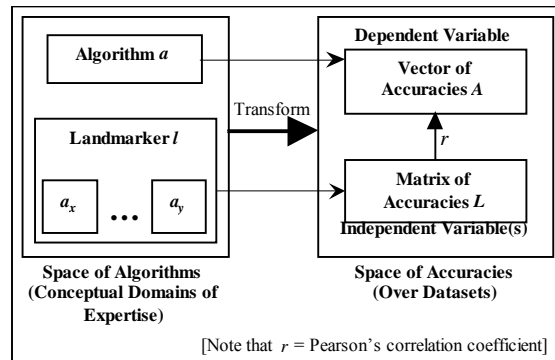


Figure 1. Operationalising $a.l$ – an example

2.2 Identifying Potential Landmarking Algorithms

Essentially, any learning algorithm could be utilised as a landmarking algorithm. Therefore, the initial search space for landmarks is extremely large. One simple method is to only consider algorithms that are less complex versions of the candidate algorithms. Such modifications may be categorised into two groups:

- **Algorithm specific reductions**, which relate to the actual inner workings of one particular algorithm. This includes: limiting the structure formed (e.g. decision stumps for decision trees), and limiting the internal search mechanisms within the algorithm (e.g. by employing randomness [5]).
- **Algorithm generic reductions**, which relate to generic modifications that may be applied to any learning algorithm. As done in [6], such modifications could be similar to the sub-sampling techniques used in ensemble literature [5].

Another method of doing this, which we first proposed in [12], is to take the set of candidate algorithms itself. This is described in further detail in the subsequent section.

3. The Proposed Hill-climbing Landmarker Generation Algorithm

In previous work [10-12], we reinterpret the role of landmarks and proposed to use a subset of the available algorithms as landmarking algorithms. More specifically, given a set of algorithms $A = \{a_1, \dots, a_n\}$, we seek to select a set of landmarks $L' = \{l_1', \dots, l_n'\}$, where the l_i' is the landmarker for algorithm a_i , $l_i' \subset A$, and the union of all $l_i' \in L'$ (written as $\text{union}(L')$) is a (proper) subset of A . Given this set of potential landmarking algorithms, we proposed a brute-force or all-subsets landmarker generation algorithm that essentially generates L' by evaluating a heuristic over the linear regression functions formed using all possible

subsets of A . This method required the computation of $n \cdot (2^{n-1} - 1)$ linear regression functions, where $n = |A|$. Given that the computation of each regression function has complexity $O(m)$, where m is the number of instances (i.e. in our case performance estimates, or datasets), the complexity of the brute-force landmarker generation algorithm is $O(nm \cdot 2^n)$, which makes the method unwieldy for high n . (However, note that this complexity is associated with training time and not runtime execution – i.e. application on a new dataset) As a more efficient alternative, we propose a hill-climbing version of the landmarker generation algorithm, which relates to the previous version of the algorithm in the same manner that the standard forward selection [14] relates to all-subsets regression [14].

The standard forward selection procedure [14] begins by assigning no independents, i.e. using only the mean value of the dependent for estimation. It then iteratively attempts to add independents one by one. In each iteration, it first computes the regression functions that each utilise a set of independents consisting of the union of the current set of chosen independents and one of the remaining (potential) independents. Subsequently, the best model *new_model* is selected using some heuristic or criterion measure (e.g. r^2 , C_p), and is compared with the previous model (i.e. the model including all the previously selected independents) *old_model* via a F -test. This F -test checks if the computed $F = (SSE(\text{old_model}) - SSE(\text{new_model})) / MSE(\text{new_model})$ is greater than 4.0 (which is roughly the F value associated with *confidence* = 0.95, $dfn = 1$, and dfd (i.e. $|S| - |A| - 1$) > 10). If this inequality holds, then: (i) the new model is adopted, (ii) the newly applied independent is removed from the potential set of independents, and (iii) we proceed to the next iteration. Else, the procedure stops and the final adopted model is *old_model*. For more details on the forward selection procedure see [14].

Essentially, the standard forward selection procedure restricts the amount of computation by excluding models that do not include the already chosen independents. More specifically, given a total number of independents n , in any iteration i , only $n - (i - 1)$ regression functions need be constructed and evaluated, each pertaining to the union of the previously chosen independents and one of the remaining independents in n that had not been chosen in any previous iteration.

Thus, instead of evaluating all possible subsets of A and computing $n \cdot (2^{n-1} - 1)$ regression functions, the new hill-climbing landmarker generation algorithm at most requires the computation of $n + \sum_{x=1}^{n-1} (x^2 + x)$ regression functions. This translates to a (training time) complexity of $O(mn^3)$, reducing a previously exponential complexity to a polynomial one. However, this is a hill-climbing approach, and thus may not find the optimal model with respect to the chosen criterion, e.g. r^2 .

3.1 Algorithm Description

The hill-climbing (i.e. forward selection) version of the landmarker generation algorithm is essentially a slightly more complicated version of the standard forward selection procedure described in the previous section; it is described in Figure 2. As per the previous all-subsets version of the algorithm, the general idea is to use a subset of the available algorithms as the components of a landmarker for each of these algorithms. Loosely, this idea assumes that: (i) several of the available algorithms will have similar domains of expertise, such that we need only use one of these domains to predict several; and (ii) if a given candidate algorithm has a very dissimilar domain of expertise (as compared to the other algorithms), that domain of expertise can be correlated to the conjunction of several others. To re-iterate, more specifically, given a set of candidate algorithms $A = \{a_1, \dots, a_n\}$, we seek to select a set of landmarkers $L' = \{l_1', \dots, l_n'\}$, such that each l_i' corresponds to the landmarker for algorithm a_i , where $l_i' \subset A$, and the union of all $l_i' \in L'$ (written as $union(L')$) is a (proper) subset of A . For example, given $A = \{a_1, a_2, a_3, a_4\}$, a possible set of selected landmarkers may be $L' = \{l_1' = \{a_1\}, l_2' = \{a_2\}, l_3' = \{a_1, a_2\}, l_4' = \{a_2\}\}$.

As in the all-subsets version, we require that $union(L') \subset A$ because our objective is to incur less computational cost than actually evaluating A ; without this condition, there is a chance that $union(L') = A$, which would invalidate the efficiency criterion. Referring back to the example above, where $L' = \{l_1' = \{a_1\}, l_2' = \{a_2\}, l_3' = \{a_1, a_2\}, l_4' = \{a_2\}\}$, we see that to obtain the performance of each $a_i \in A$, on some new dataset s_{new} , we only have to evaluate a_1 and a_2 on s_{new} (i.e. we evaluate a_1 and a_2 , and use those values to estimate the performance of a_3 and a_4). Thus, if $union(L') = A$, we would not have made any gains in efficiency, as all $a_i \in A$ would have to be evaluated. This means that in any iteration of the hill-climbing algorithm, we must ensure that the union of the current landmarkers selected must not equal A .

It also becomes evident that any a_i utilised by a landmarker (i.e. $a_i \in union(L')$), must be evaluated, and therefore need not be estimated. In fact, from our example, we see that l_1' and l_2' are really not landmarkers; they are merely evaluations of the algorithms a_1 and a_2 . This property of the problem fits nicely into the hill-climbing landmarker generation approach. It means that in any iteration:

1. Two subsets of A may be defined: (i) the algorithms to be estimated $I = \{a_i \mid a_i \subset A \setminus union(\text{current_}L')\}$ (i.e. the algorithms that require landmarkers, and correspondingly, the algorithms yet to be chosen – potential independents), and (ii) the algorithms to be evaluated $J = \{a_i \mid a_i \subset union(\text{current_}L')\}$ (i.e. the algorithms chosen to be utilised by the landmarkers – this is represented by the variable *chosen* in the pseudo code in Figure 2). Based on L' from our example, these are: $I = \{a_3, a_4\}$, and $J = \{a_1, a_2\}$.
2. We seek to move one of the algorithms from I to J . Note that essentially in any number of algorithms (less that

$|A|$) may be moved via a single iteration. However, doing this increases the complexity of the algorithm. In fact, when attempting to move $|A| - 1$ algorithms in a single iteration, we are actually executing the original all-subsets version of the algorithm.

Inputs:

- $A = \{a_1, \dots, a_n\}$, a set of n candidate algorithms; and
- for $\forall a_i \in A$, $\varphi_{a_i} = \{\varphi_{a_i,1}, \dots, \varphi_{a_i,m}\}$, the set of performance estimates for algorithm a_i on a corresponding set of datasets $S = \{s_1, \dots, s_m\}$.

Output:

- $L' = \{l_1', \dots, l_n'\}$, a set of landmarks, where each l_i' is the chosen landmark for a_i .

Let:

- $get_regression_fn(dependent, independents)$ returns the **coefficients** of the for the linear regression function corresponding to the input **dependent** and **independents**.
- $find_heuristic(coefficients)$ returns the heuristic (e.g. r^2 , C_p) for the regression function corresponding to **coefficients**.
- $SSE(coefficients)$ returns the sum of squared errors associated with the regression function corresponding to **coefficients**.
- $MSE(coefficients)$ returns the mean squared error associated with the regression function corresponding to **coefficients**.
- $\forall a_i \in A$, $\varphi_{a_i} = \{\varphi_{a_i,1}, \dots, \varphi_{a_i,m}\}$ be globally accessible
- $max_independents$ = maximum number of algorithms to evaluate.

The hillclimbing landmarker generation algorithm:

```

generate_landmarkers(A) returns L'
1 L' = nil
2 for  $\forall a_i \in A$ 
3    $L[a_i] = get\_regression\_fn(a_i, \emptyset)$ 
4   chosen =  $\emptyset$ 
5   valid = true
6   while valid
7     fns = nil
8     sum_heuristic = nil
9     for  $\forall a_j \in A \setminus chosen$ 
10      for  $\forall a_k \in A \setminus \{chosen \cup a_j\}$ 
11         $fns[a_i][a_j] = get\_regression\_fn(a_i, chosen \cup a_j)$ 
12         $sum\_heuristic[a_j] += find\_heuristic(fns[a_i][a_j])$ 
13      best_independent =  $a_k \mid sum\_heuristic[a_k] =$ 
            $\underset{\forall a_x \in A \setminus chosen}{\operatorname{argmax}} sum\_heuristic[a_x]$ 
14      chosen = chosen  $\cup$  best_independent
15      valid_update = false
16      for  $\forall a_i \in A$ 
17        if  $a_i \in chosen$ 
18           $L'[a_i] = a_i$ 
19        else if [ $SSE(L'[a_i]) - SSE(fns[a_i][best\_independent])$ ] /
            $MSE(fns[a_i][best\_independent]) > 4.0$ 
20           $L'[a_i] = fns[a_i][best\_independent]$ 
21          valid_update = true
22      if !valid_update or |chosen| == max_independents
23        valid = false

```

Figure 2. Pseudo code for the proposed hill-climbing landmarker generation algorithm

With either version of the landmarker generation algorithm, we begin with $I = A$ and $J = \emptyset$. To shift $k < |A|$ algorithms from I to J , we essentially select the k algorithms that yield the greatest overall utility (i.e.

measured based on the heuristic chosen – e.g. r^2 or $r^2 + efficiency_gained$). As an independent (i.e. predictor variable), each algorithm attains a certain amount of correlation to each of its applicable dependents. Thus, we gauge the utility of each independent based on the mean correlation with its applicable dependents (note that the number of applicable dependents for any potential independent will always be the same; i.e. in iteration i , there will be $|A| - |J| - 1$ dependents for each independent). For example, given $A = \{a_1, a_2, a_3, a_4\}$, and assuming we wish choose a single algorithm ($k = 1$) based solely on r^2 , then given the r^2 values for A in Table 1, we see that as an independent (i.e. a predictor based on linear regression), a_4 has the highest overall utility. The calculation for cases where $k > 1$ is slightly more complicated, and since the proposed hill-climbing version only requires the calculation for $k = 1$, we will not elaborate further; for a description of this selection method for $k > 1$, see [12].

Independent	Dependent				Mean r^2
	a_1	a_2	a_3	a_4	
a_1	NA	0.5	0.3	0.7	$1.5 / 3 = \mathbf{0.5}$
a_2	0.5	NA	0.7	0.6	$1.8 / 3 = \mathbf{0.6}$
a_3	0.3	0.7	NA	0.8	$1.8 / 3 = \mathbf{0.6}$
a_4	0.7	0.6	0.8	NA	$2.1 / 3 = \mathbf{0.7}$

Table 1. Example r^2 values and overall utility of $A = \{a_1, a_2, a_3, a_4\}$ for the one algorithm case.

Another issue that arises when adapting the standard forward selection to our landmarker generation scenario is that the original stopping criterion (i.e. F -test) must be modified. Essentially, the new stopping criteria requires that: (i) the F -test be adapted to the case where a single independent is chosen for multiple dependents, and (ii) that a limit be imposed on the total number of independents so that $union(L') \subset A$ is ensured (and so that a dynamic limit on the training time computational cost may be imposed).

Primarily, the proposed algorithm seeks to mimic the standard forward selection, which in each iteration selects the independent with the highest criterion measure (e.g. r^2). However, since the proposed algorithm makes this selection based on an overall utility (i.e. criterion measure over all remaining dependents), it is likely that at least for some dependents, the independent with the highest criterion measure is not chosen. To account for this (partially), we do not stop attempting to utilise more independents for any remaining dependent even when an F -test for that dependent fails. Instead, our stopping criterion (over all remaining dependents) applies whenever: (i) the selected independents J reaches a certain limit – i.e. $|J| >$ the maximum number of algorithm we wish to evaluate; and (ii) if all the F -tests (i.e. over all remaining dependents) fail for the currently selected independent.

3.2 Heuristics for Correlativity and Efficiency

To choose between the various potential landmarkers (i.e. the regression functions corresponding to the various independent set options in each iteration), we require a criterion measure or heuristic that measures the utility of each remaining dependent (i.e. $\forall a_i \in A \setminus J$) to potential independent set (i.e. $(\forall a_i \in A \setminus (I \cup a_i)) I \cup a_i$) pairing – i.e. *find_heuristic(.)* on Line 12 of Figure X. Given the specified correlativity and efficiency landmarker criteria, two heuristics are evaluated:

1. The squared Pearson product moment coefficient or coefficient of determination r^2 . By adopting this basic measure of correlation (to satisfy the correlativity criterion), we rely on the stopping criteria $union(L') \subset A$ to ensure the efficiency criterion is met. However, doing so means that in each iteration, the choice of independent is purely guided by its correlativity utility – i.e. the (computational) cost over the potential independents are ignored.
2. A semi cost-sensitive variant of the plain r^2 criterion: $r^2(a_i, l_k) + ((eff(A) - eff(l_k)) / eff(A))$, where a_i corresponds to the dependent, l_k to the potential landmarker (i.e. set of independents) and A to the set of all candidate algorithms, so that $r^2(a_i, l_k)$ is the r^2 value observed over the $a_i - l_k$ pairing, and $eff(.)$ is the estimated overall computational cost (i.e. the time taken for training and testing) of its subject. Essentially, $(eff(A) - eff(l_k)) / eff(A)$ corresponds to the amount of efficiency gained (or time/computation saved) when $l_k \subset A$ is evaluated instead of A . Note that we have proposed several approaches for estimating $eff(.)$ [10, 11]. In this paper, we adopt the simpler and less space intensive method specified in [11], which simply takes the mean training and runtime computational cost over the training datasets for each relevant a_x .

4. Experiments, Results and Analysis

For our experiments we utilise 10 classification learning algorithms from WEKA [19] (i.e. naïve Bayes, k -nearest neighbour (with $k = 1$ and 7), support vector machine, decision stump, J4.8 (a WEKA implementation of C4.5), random forest, decision table, Ripper, and ZeroR) and 34 classification datasets randomly chosen from the UCI repository [2]. To evaluate the accuracy of each candidate algorithm on each dataset, stratified ten-fold cross-validation was employed. The effectiveness of the proposed landmarker generation algorithm is evaluated using the leave-one-out cross-validation approach. This corresponds to n -fold cross-validation, where n is the number of instances, which in our case is 34, each pertaining to one UCI dataset.

For each fold we use 33 of the datasets to generate landmarkers as described in Section 3. The resultant set of landmarkers indicates which algorithms must be evaluated

and which will be estimated (i.e. the algorithms in $union(L')$, and the remaining $A \setminus union(L')$ respectively). On the dataset left out, we first run the algorithms that must be evaluated and then use their accuracy results to estimate the performance of the other algorithms using the regression functions computed during landmarker generation.

The version of the algorithm described in Section 3.1 will attempt to find a set of landmarkers L' such that $|union(L')| \leq max_independents$ (the user specified parameter) and where the chosen landmarkers only includes algorithms (i.e. independents) that have not failed all F -tests. Thus it is possible that the hill-climbing landmarker generation algorithm returns an L' such that $|union(L')| < max_independents$. Correspondingly, some modifications to the proposed algorithm were made in order to make a direct comparison with the results from the all-subsets version, which (in those experiments – see [10-12]) produces a set of landmarkers for each possible size of $|union(L')| < |A|$. In order to ensure that a specific number of independents are utilised (i.e. a specific number of algorithms are run), the hill-climbing version will when it encounters the situation where $|union(L')| < max_independents$ and all F -tests for the current independent fail, continue to move algorithms from I to J without changing the chosen models (i.e. independents) for the remaining algorithm to be estimated in I . In each such iteration, the dependent algorithm (i.e. landmarked algorithm) with the model with lowest utility is moved.

This leaves us with 9 sets of accuracy *estimates* for each of the candidate algorithms over each of the UCI datasets. Three evaluations are performed over the accuracy estimates:

- Efficiency gained (*EG*): for each held-out dataset and landmarker set size, we compute the percentage of computation saved by employing the landmarker. This saving is the portion of the computational time incurred by conducting ten-fold cross-validation over all the candidate algorithms that is saved by instead running only the algorithms associated with the landmarker in question. For each landmarker set size, we report the mean *EG* recorded over all datasets.
- Rank order correlation (r_s): for each held-out dataset and landmarker set size, we utilise the Spearman's rank order correlation coefficient r_s , to determine the correlation between: (i) the rank order of the accuracies estimated via the landmarkers and regression, and (ii) the rank order of the accuracies evaluated via ten-fold cross-validation. For each landmarker set size, we report the mean r_s recorded over all datasets.
- Algorithm-pair ordering (*AO*): for each dataset and landmarker set size, we compare the order of each pair of algorithms (e.g. if $acc(a_1) > acc(a_2)$) based on the estimated (via the landmarkers and regression) and evaluated accuracies (via ten-fold cross-validation). For each landmarker set size, we report the mean (across all datasets) of the percentage of pairings in which the order is predicted correctly. Note that there

are $^{10}C_2 = 45$ algorithms pairings with 10 algorithms. However, one notices that when all 10 algorithms are employed by the set of landmarkers, no landmarkers are required, and we are simply performing ten-fold cross-validation. Accordingly, for a landmarker set size of x , those x algorithms are evaluated, not estimated. Thus, $^x C_2$ algorithm pairs will correspond to the ordering that is found via ten-fold cross-validation. We denote this as assured AO , which is the accuracy associated with pairings that are guaranteed to be correct.

	No. Algorithms Evaluated, $ \text{union}(L') = J = A - I $								
	1	2	3	4	5	6	7	8	9
Mean EG	92.4	85.3	83.6	83.1	82.7	44.1	52.1	23.8	8.4
Mean r_s	0.55	0.59	0.68	0.73	0.79	0.81	0.86	0.92	0.97
Mean AO	71.5	74.1	78.0	80.5	82.9	85.2	88.7	93.1	96.3
Assured AO	0.0	2.2	6.7	13.3	22.2	33.3	46.7	62.2	80.0
Mean r^2	0.64	0.81	0.89	0.92	0.94	0.95	0.96	0.97	0.98

Table 2. Results for the all-subsets version (plain r^2 heuristic)

	No. Algorithms Evaluated, $ \text{union}(L') = J = A - I $								
	1	2	3	4	5	6	7	8	9
Mean EG	97.5	97.1	97.0	88.9	82.7	82.1	67.5	54.2	9.2
Mean r_s	0.52	0.59	0.69	0.71	0.76	0.83	0.87	0.91	0.93
Mean AO	70.7	73.8	78.4	79.7	82.8	86.7	89.4	92.4	94.8
Assured AO	0.0	2.2	6.7	13.3	22.2	33.3	46.7	62.2	80.0
Mean r^2	0.63	0.81	0.87	0.88	0.89	0.91	0.92	0.92	0.93

Table 3. Results for the all-subsets version (r^2 plus efficiency gained heuristic)

	No. Algorithms Evaluated, $ \text{union}(L') = J = A - I $								
	1	2	3	4	5	6	7	8	9
Mean EG	92.4	92.4	84.7	84.2	83.8	68.7	51.5	23.5	9.0
Mean r_s	0.53	0.47	0.57	0.73	0.79	0.83	0.87	0.92	0.96
Mean AO	70.8	68.6	72.6	80.0	82.8	86.1	89.4	92.8	96.0
Assured AO	0.0	2.2	6.7	13.3	22.2	33.3	46.7	62.2	80.0
Mean r^2	0.64	0.78	0.85	0.91	0.92	0.93	0.94	0.94	0.95

Table 4. Results for the hill-climbing version (plain r^2 heuristic)

	No. Algorithms Evaluated, $ \text{union}(L') = J = A - I $								
	1	2	3	4	5	6	7	8	9
Mean EG	97.5	97.1	96.7	96.6	88.3	82.0	67.1	49.9	10.6
Mean r_s	0.50	0.56	0.74	0.74	0.77	0.84	0.87	0.89	0.94
Mean AO	69.9	72.6	80.3	80.7	83.4	86.3	89.2	91.7	95.4
Assured AO	0.0	2.2	6.7	13.3	22.2	33.3	46.7	62.2	80.0
Mean r^2	0.63	0.80	0.85	0.87	0.89	0.91	0.91	0.92	0.93

Table 5. Results for the hill-climbing version (r^2 plus efficiency gained heuristic)

The results of the all-subsets version of the landmarker generation algorithm using the plain r^2 and r^2 plus efficiency gained heuristics are given in Table 2 and 3 respectively, while the results from the hill-climbing (forward selection) version are given in Tables 4 and 5.

From Tables 2 through 5, we observe that the hill-climbing landmarker generation algorithm is quite closely matches the predictive performance of the all-subsets version. Correspondingly, we find that even when few landmarking algorithms are utilised, the generated landmarkers are still able to produce a reasonable estimate while making a sizable gain in efficiency. And as with the all-subsets result, when we allow the generation algorithm to utilise larger sets of candidate algorithms as landmarkers (i.e. as we allow larger $|\text{union}(L')|$), the r^2 , r_s , and AO all increase, and the efficiency gained decreases, with all the values approaching the ten-fold cross-validation result.

Acknowledgements

The authors would like to acknowledge the support of the Smart Internet Technology CRC in this research.

References

1. Bensusan, H. (1998). *God Doesn't Always Shave with Occam's Razor: Learning When and How to Prune*. In Proceedings of the *European Conference on Machine Learning*, 119-124.
2. Blake, C., and Merz, C. (1998). *UCI Repository of Machine Learning Databases*. University of California, Irvine, Department of Information and Computer Sciences.
3. Brazdil, P., Gama, J., and Henery, R. (1994). *Characterising the Applicability of Classification Algorithms Using Meta Level Learning*. In Proceedings of the *European Conference on Machine Learning*, 84-102.
4. Brazdil, P., Soares, C., and Costa, J. (2003). *Ranking Learning Algorithms: Using IBL and Meta-learning on Accuracy and Time Results*. *Machine Learning*, 50(3), 251-277.
5. Dietterich, T. (1997). *Machine Learning Research: 4 Current directions*. *Artificial Intelligence Magazine*, 18(4), 97-136.
6. Fürnkranz, J., and Petrak, J. (2001). *An Evaluation of Landmarking Variants*. In Proceedings of the *European Conference on Machine Learning, Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-learning*, 57-68.
7. Giraud-Carrier, C., Vilalta, R., and Brazdil, P. (2004). *Introduction to the Special Issue on Meta-Learning*. *Machine Learning*, 54(3), 187-193.

8. Kalousis, A., and Hilario, M. (2001). *Model Selection via Meta-learning*. In Proceedings of the *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 222-233.
9. Kalousis, A., and Hilario, M. (2001). *Model Selection via Meta-learning: A Comparative Study*. *International Journal on Artificial Intelligence Tools*, 10(4), 525-554.
10. Ler, D., Koprinska, I., and Chawla, S. (2004). *Comparisons between Heuristics Based on Correlativity and Efficiency for Landmarker Generation*. In Proceedings of the *4th International Conference on Hybrid Intelligent Systems (HIS-04)*, (In Press).
11. Ler, D., Koprinska, I., and Chawla, S. (2004). *A Landmarker Selection Algorithm Based on Correlation and Efficiency Criteria*. In Proceedings of the *17th Australian Joint Conference on Artificial Intelligence (AI-04)*, (In Press).
12. Ler, D., Koprinska, I., and Chawla, S. (2004). *A New Landmarker Generation Algorithm Based on Correlativity*. In Proceedings of the *2004 International Conference on Machine Learning and Applications (ICMLA-04)*, (In Press).
13. Michie, D., Spiegelhalter, D., and Taylor, C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.
14. Miller, A. (2002). *Subset Selection in Regression* (2nd ed). Chapman and Hall/CRC.
15. Pfahringer, B., Bensusan, H., and Giraud-Carrier, C. (2000). *Meta-learning by Landmarking Various Learning Algorithms*. In Proceedings of the *International Conference on Machine Learning*, 743-750.
16. Schaffer, C. (1993). *Technical Note: Selecting a Classification Method by Cross-validation*. *Machine Learning*, 13(1), 135-143.
17. Vilalta, R., and Drissi, Y. (2002). *A Prespective View and Survey of Meta-learning*. *Artificial Intelligence Review*, 18(2), 77-95.
18. Wickens, T. (1995). *The Geometry of Multivariate Statistics*. LEA Publishers.
19. Witten, I., and Frank, E. (2000). *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann.
20. Wolpert, D. (2001). *The Supervised Learning No-free-lunch Theorems*. In Proceedings of the *Soft Computing in Industry - Recent Applications*, 25-42.