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

Daren Ler, Irena Koprinska, and Sanjay Chawla

School of Information Technologies, University of Sydney
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. Recently, we proposed a new algorithm for making such a selection based on landmarking - a meta-learning strategy that utilises meta-features that are measurements based on efficient learning algorithms. This algorithm, which creates a set of landmarkers 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 has exponential computational complexity for training. 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 landmarkers 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 (Wolpert, 2001), as well as empirical results (Michie et al., 1994) suggest that no one algorithm is generically superior; this means that we cannot always rely on one particular algorithm to solve all our learning problems. Accordingly, selection is typically done via some form of holdout testing. 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 (Giraud-Carrier et al., 2004; Vilalta and Drissi, 2002) techniques utilise past experience or meta-knowledge 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 (e.g. Kalousis and Hilario, 2001; Brazdil et al., 1994; Michie et al., 1994), there has been no consensus on what constitutes good features.

Landmarking (Fürnkranz and Petrak, 2001; Pfahringer et al., 2000) is a new and promising approach that characterises datasets by directly measuring the performance of simple and fast learning algorithms called landmarkers. However, the selection of landmarkers is typically done in an ad hoc fashion, with the landmarkers generated focused on characterising an arbitrary set of algorithms. In previous work (Ler et al., 2004b, 2004c), we reinterpreted the role of landmarkers, defining each as a set of learning algorithms that characterises the pattern of performance of one specific learning algorithm. As criteria, we specified that these landmarkers 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 (Miller, 2002)) has an exponential training time computational complexity, making it unwieldy.

In this paper, we propose an alternate hill-climbing landmarker generation algorithm based on the forward selection (Miller, 2002) 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.

2. Landmarking and Landmarkers

In the literature (Fürnkranz and Petrak, 2001; Pfahringer et al., 2000), 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. However, in previous Landmarking work (Pfahringer et al., 2000), despite the presence of two landmarker criteria (i.e. efficiency and bias diversity), no actual mechanism for generating appropriate landmarkers were defined, and the choice of landmarkers was made in an ad hoc fashion.

In (Ler et al., 2004b, 2004c), landmarking and landmarkers were redefined. Let: (i) a *landmarking element* be an algorithm whose performance is utilised as a dataset characteristic, and (ii) a *landmarker* be 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 – i.e. landmarkers.

Consequently, we also suggest new landmarker criteria. Previously (Pfahringer et al., 2000), it was suggested that each landmarking element be (i) efficient, and (ii) bias diverse with respect to the other chosen landmarkers. However, as landmarker (and not landmarking element) criteria, there is no guarantee that the landmarkers selected via these criteria will be able to characterise the performance of one, and even less so, a set of candidate algorithms (Ler et al., 2004b). Instead, in (Ler et al., 2004b, 2004c), we proposed that each landmarker be: (i) correlated – i.e. each landmarker should as closely as possible have its output resemble the performance of the candidate algorithm being landmarked, and (ii) efficient.

2.1 Using Criteria to Select Landmarkers

Intuitively, two algorithms with similar performance will be closer to each other in a space of algorithm performance. Conceptually, the distance between two algorithms $\boldsymbol{a}$ and $\boldsymbol{l}$ can be regarded as $||\boldsymbol{a} - \boldsymbol{l}||$. Also, we may express $||\boldsymbol{a} - \boldsymbol{l}||^2$ as $||\boldsymbol{a}||^2 + ||\boldsymbol{l}||^2 - 2\,\boldsymbol{a}.\boldsymbol{l}$. Thus, if $\boldsymbol{a}$ is close to $\boldsymbol{l}$, this implies that $||\boldsymbol{a} - \boldsymbol{l}||^2$ is small, and thus that $\boldsymbol{a}.\boldsymbol{l}$ is relatively large. This pertains to the correlativity criterion we use to check if a landmarker (e.g. $\boldsymbol{l}$) is representative of the performance of some candidate algorithm (e.g. $\boldsymbol{a}$). Accordingly, correlation may be measured based on r (Wickens, 1995):

$$r = \cos\angle(\boldsymbol{a},\boldsymbol{l}) = \frac{\boldsymbol{a}.\boldsymbol{l}}{||\,\boldsymbol{a}\,||\;||\,\boldsymbol{l}\,||} = \frac{\mathrm{cov}(\boldsymbol{a},\boldsymbol{l})}{\sigma_{\boldsymbol{a}}\;\sigma_{\boldsymbol{l}}}$$

Several other heuristics similarly apply, including r^2 (i.e. the squared Pearson's correlation coefficient), Mallow's C_p criterion, Akaike and Bayesian Information Criteria (i.e. *AIC* and *BIC* respectively) (Miller, 2002). 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 (Ler et al., 2004a).

2.2 Potential Landmarking Elements

Essentially, any learning algorithm could be utilised as a landmarking element. Therefore, the initial search space for landmarking elements is extremely large. One remedy 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, and could include: limiting the structure formed (e.g. decision stumps for decision trees), and/or limiting the internal search mechanisms within the algorithm (e.g. using randomness (Dietterich, 1997)).
- **Algorithm generic reductions**, which relate to generic modifications that may be applied to any learning algorithm. As per [4], such modifications could be similar to the sub-sampling techniques used in ensemble literature (Dietterich, 1997).

Another method of doing this, is to utilise a subset set of the given candidate algorithms itself (Ler et al., 2004b, 2004c). This is described in further detail in Section 3.

3. The Proposed Hill-climbing Landmarker Generation Algorithm

In previous work (Ler et al., 2004a, 2004b, 2004c), we reinterpret the role of landmarkers and proposed to use a subset of the available algorithms as landmarking elements. More specifically, given a set of candidate algorithms $\boldsymbol{A} = \{\boldsymbol{a_1}, \ldots, \boldsymbol{a_n}\}$, we seek to select a set of landmarkers $\boldsymbol{L'} = \{\boldsymbol{l_1'}, \ldots, \boldsymbol{l_n'}\}$ (where each $\boldsymbol{l_i'}$ is the landmarker for $\boldsymbol{a_i}$ and utilises the landmarking elements $\boldsymbol{l_i'}.\mathit{elements}$) such that: (i) $\forall \boldsymbol{l_i'} \in \boldsymbol{L'}$, $\boldsymbol{l_i'}.\mathit{elements} \subset \boldsymbol{A}$, and (ii) the union of all $\boldsymbol{l_i'}.\mathit{elements}$ (i.e. $\boldsymbol{L'}.\mathit{elements}$) is a (proper) subset of $\boldsymbol{A}$. Thus, given the potential landmarking element sets (i.e. each subset of $\boldsymbol{A}$), the all-subsets landmarker generation algorithm would generate $\boldsymbol{L'}$ by selecting the landmarking element set (i.e. the subset of $\boldsymbol{A}$) that achieved the highest heuristic score. This method required the computation of $\sum_{i=1}^{n-1} \binom{n}{i} (n-i)$ linear regression functions, where $\boldsymbol{n} = |\boldsymbol{A}|$. Given that the computation of each regression function has complexity $O(\boldsymbol{m})$, where $\boldsymbol{m}$ is the number of performance estimates (i.e. datasets), the complexity of the all-subsets landmarker generation algorithm is $O(\boldsymbol{nm}.2^{\boldsymbol{n}})$, which makes the method unwieldy for high $\boldsymbol{n}$. (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 (Miller, 2002) relates to all-subsets regression (Miller, 2002).

The standard forward selection procedure begins by assigning no independents – i.e. using only the mean value of the dependent for estimation. It then iteratively attempts to add landmarking elements (i.e. 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 unused 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 $\boldsymbol{F}$ = (SSE(***old_model***) - SSE(***new_model***)) / MSE(***new_model***) is greater than 4.0 (which is roughly the F value associated with *confidence* = 0.95, *dfn* = 1, and *dfd* (i.e. $\boldsymbol{n} - \boldsymbol{m} - 1$) > 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 (Miller, 2002).

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 $\boldsymbol{n}$, in any iteration $\boldsymbol{i}$, only $\boldsymbol{n}$ - ($\boldsymbol{i}$ - 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 $\boldsymbol{n}$ that had not been chosen in any previous iteration.

Thus, instead of evaluating all possible subsets of $\boldsymbol{A}$ and computing $\sum_{i=1}^{n-1}\binom{n}{i}(n-i)$ regression functions, the new hill-climbing landmarker generation algorithm at most requires the computation of $\sum_{i=1}^{n-1}\binom{n-i-1}{1}.(n-i)$ 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 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 1. As per the previous all-subsets version of the algorithm, the general idea is to use a subset of the available algorithms as potential landmarking elements. Loosely, this idea assumes that: (i) several of the available algorithms will have similar performance patterns, such that one pattern is representative of several; and (ii) if a given algorithm has a very dissimilar performance pattern (as compared to the other algorithms), that performance pattern can be correlated to the conjunction of several others. To re-iterate, given a set of candidate algorithms $\boldsymbol{A} = \{\boldsymbol{a_1}, \ldots, \boldsymbol{a_n}\}$, we seek to select a set of landmarkers $\boldsymbol{L'} = \{\boldsymbol{l_1'}, \ldots, \boldsymbol{l_n'}\}$ (where each $\boldsymbol{l_i'}$ is the landmarker for $\boldsymbol{a_i}$ and utilises the landmarking elements $\boldsymbol{l_i'}.elements$) such that: (i) $\forall \boldsymbol{l_i'} \in \boldsymbol{L'}$, $\boldsymbol{l_i'}.elements \subset \boldsymbol{A}$, and (ii) the union of all $\boldsymbol{l_i'}.elements$ (i.e. $union(\boldsymbol{L'}.elements)$) is a (proper) subset of $\boldsymbol{A}$. For example, given $\boldsymbol{A} = \{\boldsymbol{a_1}, \boldsymbol{a_2}, \boldsymbol{a_3}, \boldsymbol{a_4}\}$, a possible set of selected landmarkers may correspond to $\boldsymbol{L'}$, such that $\boldsymbol{L'}.elements = \{\boldsymbol{a_1}, \boldsymbol{a_2}\}$, and $\boldsymbol{l_1'}.elements = \{\boldsymbol{a_1}\}$, $\boldsymbol{l_2'}.elements = \{\boldsymbol{a_2}\}$, $\boldsymbol{l_3'}.elements = \{\boldsymbol{a_1}, \boldsymbol{a_2}\}$, $\boldsymbol{l_4'}.elements = \{\boldsymbol{a_2}\}$.

As in the all-subsets version, we require that $\boldsymbol{L'}.elements \subset \boldsymbol{A}$ because our objective is to incur less computational cost then actually evaluating $\boldsymbol{A}$; without this condition, there is a chance that $\boldsymbol{L'}.elements = \boldsymbol{A}$, which would invalidate the efficiency criterion. Referring back to the example above, we see that to obtain the performance of each $\boldsymbol{a_i} \in \boldsymbol{A}$, on some new dataset $\boldsymbol{s_{new}}$, we only have to evaluate $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$ on $\boldsymbol{s_{new}}$ (i.e. we evaluate $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$, and use those values to estimate the performance of $\boldsymbol{a_3}$ and $\boldsymbol{a_4}$).

Thus, if $\boldsymbol{L'}.elements = \boldsymbol{A}$, we would not have made any gains in efficiency, as all $\boldsymbol{a_i} \in \boldsymbol{A}$ 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 landmarking algorithms selected must not equal $\boldsymbol{A}$. (Note that throughout the remainder of this paper, we use: the term landmarking elements interchangeably with the term independents, and the term dependents for the remaining algorithms – i.e. $\boldsymbol{A} \setminus$ independents.)

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

Output:
- $\boldsymbol{L'} = \{\boldsymbol{l_1'}, \ldots, \boldsymbol{l_n'}\}$, a set of landmarkers, where each $\boldsymbol{l_i'}$ is the chosen landmarker for $\boldsymbol{a_i}$.

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) 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***.
- $\forall \boldsymbol{a_i} \in \boldsymbol{A}$, $\varphi_{\boldsymbol{a_i}} = \{\varphi_{\boldsymbol{a_i},1}, \ldots, \varphi_{\boldsymbol{a_i},\boldsymbol{m}}\}$ 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 ∀a_i ∈ A
3     L[a_i] = get_regression_fn(a_i, ϕ)
4  chosen = ϕ
5  valid = true
6  while valid
7     fns = nil
8     sum_heuristic = nil
9     for ∀a_i ∈ A \ chosen
10       for ∀a_j ∈ A \ {chosen ∪ a_i}
11          fns[a_i][a_j] = get_regression_fn(a_i, chosen ∪ a_j)
12          sum_heuristic[a_j] += find_heuristic(fns[a_i][a_j])
13    best_independent = a_k | sum_heuristic[a_k] =
                 argmax_{∀a_x ∈ A \ chosen}  sum_heuristic[a_x]
14    chosen = chosen ∪ best_independent
15    valid_update = false
16    for ∀a_i ∈ A
17    if a_i ∈ 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 1. Pseudo code for the proposed hill-climbing landmarker generation algorithm

It also becomes evident that any chosen landmarking element a_i, 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. Thus, with each iteration, two points become clear. The first is that two subsets of A may be defined: (i) the algorithms to be estimated $I = \{a_i \mid a_i \subset A \setminus current\_L'.elements\}$ (i.e. dependents – the algorithms that require landmarkers; and correspondingly, the algorithms yet to be chosen – the potential independents or landmarking elements), and (ii) the algorithms to be evaluated $J = \{a_i \mid a_i \subset current\_L'.elements\}$ (i.e. the chosen landmarking algorithms – represented by the variable ***chosen*** in the pseudo code in Figure 1). Based on $L'.elements$ from our example, these are: $I = \{a_3, a_4\}$, and $J = \{a_1, a_2\}$. Secondly, we seek to move one of the algorithms from I to J. Note that essentially any number of algorithms (less than $|A|$) 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.

With either version of the landmarker generation algorithm, we begin with $I = A$ and $J = \phi$. 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, 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. landmarking algorithm), 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 the selection method for $k > 1$, see (Ler et al., 2004c).

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 $L'.elements \subset A$ is ensured (so that a dynamic limit on the training time computational cost may be imposed).

The hill-climbing algorithm seeks to mimic 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, it is likely that at least for some dependents, the independent with the highest criterion measure is not chosen. To (partially) account for this, we do not stop attempting to utilise more independents for any remaining dependent even when an *F*-test for that dependent fails. Instead, out 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.

Indep.	Dependent				Mean r^2
	a_1	a_2	a_3	a_4	
a_1	NA	0.5	0.3	0.7	1.5/3 = **0.5**
a_2	0.5	NA	0.7	0.6	1.8/3 = **0.6**
a_3	0.3	0.7	NA	0.8	1.8/3 = **0.6**
a_4	0.7	0.6	0.8	NA	2.1/3 = **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.

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. $(\forall a_j \in A \setminus (I \cup a_i))\ I \cup a_j$) pairing. Given the specified correlativity and efficiency landmarker criteria, two heuristics are evaluated. The first is the squared Pearson product moment coefficient or coefficient of determination r^2. By adopting this basic measure of correlation, we rely on the stopping criteria $L'.elements \subset A$ to ensure the efficiency criterion is met. This 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 is ignored. And the second, a semi cost-sensitive variant of the plain r^2 criterion: $r^2(a_i, l_k) + ((eff(A) - eff(\underline{l_k})) / eff(A))$, where a_i is the dependent, l_k is the potential landmarker, and A is the set of all candidate algorithms, and thus $r^2(a_i, l_k)$ 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, $(eff(A) - eff(\underline{l_k})) / eff(A)$ corresponds to the amount of efficiency gained (or time/computation saved) when $l_k.elements$ is evaluated instead of A. Note that we have proposed several approaches for estimating $eff(..)$ (Ler et al., 2004a, 2004b). In this paper, we adopt the simpler and less space intensive method specified in (Ler et al., 2004b), which simply takes the mean training and runtime computational cost over the training datasets for each relevant a_x in A.

4. Experiments, Results and Analysis

For our experiments we utilise 10 classification learning algorithms from WEKA (Witten and Frank, 2000) (i.e. naïve Bayes, *k*-nearest neighbour (with k = 1 and 7), polynomial support vector machine, decision stump, J4.8 (a WEKA implementation of C4.5), random forest, decision table, Ripper, and ZeroR) (i.e as $\boldsymbol{A}$) and 34 classification datasets randomly chosen from the UCI repository (Blake and Merz, 1998). Leave-one-out cross-validation is used to test the proposed landmarker generation algorithm.

	No. Algorithms Evaluated, $\lvert \boldsymbol{L'}.elements \rvert$								
	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)

	No. Algorithms Evaluated, $\lvert \boldsymbol{L'}.elements \rvert$								
	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 + *EG*)

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 estimated (i.e. the algorithms in $\boldsymbol{L'}.elements$, and the remaining $\boldsymbol{A} \setminus \boldsymbol{L'}.elements$ respectively). On the dataset left out, we first run the algorithms that must me evaluated and then use their accuracy results (attained via stratified ten-fold cross-validation) 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 $\boldsymbol{L'}$ such that $\lvert \boldsymbol{L'}.elements \rvert \leq \boldsymbol{max\_independents}$ (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 $\boldsymbol{L'}$ such that $\lvert \boldsymbol{L'}.elements \rvert < \boldsymbol{max\_independents}$. 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 (Ler et al., 2004b)) produces a set of landmarkers for each possible size of $\boldsymbol{L'}.elements < \lvert \boldsymbol{A} \rvert$. In order to ensure that a specific number of independents are utilised (i.e. a specific number of landmarking algorithms are run), the hill-climbing version will, when it encounters the situation where $\lvert \boldsymbol{L'}.elements \rvert < \boldsymbol{max\_independents}$ and all *F*-tests for the current independent fail, continue to move algorithms from $\boldsymbol{I}$ to $\boldsymbol{J}$ without changing the chosen models (i.e. independents) for the remaining algorithm to be estimated in $\boldsymbol{I}$. In each such iteration, the dependent algorithm (i.e. landmarked algorithm) with the model with lowest utility is moved. Thus, for each dataset/fold, 9 sets of landmarkers and correspondingly, 9 sets of accuracy estimates for $\boldsymbol{A}$ are attained (since $\lvert \boldsymbol{A} \rvert = 10$). The following three evaluations are reported for each of these sets:

- **Efficiency gained** (*EG*): the mean percentage of computation saved (across the 34 datasets or folds) by running the corresponding $\boldsymbol{L'}.elements$ instead of $\boldsymbol{A}$.
- **Rank order correlation** (r_s): the mean Spearman's rank order correlation coefficient r_s (across the 34 datasets or folds), measuring the rank 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.
- **Algorithm-pair ordering** (*AO*): the mean percentage of algorithm pairings in which the order predicted via the landmarkers is the same as that observed via ten-fold cross-validation. Note, with $\lvert \boldsymbol{A} \rvert = 10$, there are ${}^{10}C_2 = 45$ pairings. Given that $\boldsymbol{x} = \lvert \boldsymbol{L'}.elements \rvert$ are evaluating (via ten-fold cross-validation), ${}^{x}C_2$ algorithm pairs will correspond to the ordering that is found via ten-fold cross-validation. We denote this as *assured AO*, the pairings that are guaranteed to be correct.

The results of the all-subsets version of the landmarker generation algorithm using the plain r^2 and r^2 + *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 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 $\lvert \boldsymbol{L'}.elements \rvert$), 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.

	No. Algorithms Evaluated, | ***L'***.*elements*|								
	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)

	No. Algorithms Evaluated, | ***L'***.*elements*|								
	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 + *EG*)

Acknowledgements

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

References

Blake, C.; and Merz, C. 1998. *UCI Repository of Machine Learning Databases*. University of California, Irvine, Department of Information and Computer Sciences.

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.

Dietterich, T. 1997. Machine learning research: 4 current directions. *Artificial Intelligence Magazine*, 18(4): 97-136.

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.

Giraud-Carrier, C.; Vilalta, R.; and Brazdil, P. 2004. Introduction to the special issue on meta-learning. Machine Learning, 54(3): 187-193.

Kalousis, A.; and Hilario, M. 2001. Model selection via meta-learning: A Comparative Study. *International Journal on Artificial Intelligence Tools*, 10(4): 525-554.

Ler, D.; Koprinska, I.; and Chawla, S. 2004a. Comparisons between heuristics based on correlativity and efficiency for landmarker generation. In *Proceedings of the 4th International Conference on Hybrid Intelligent Systems*, in press.

Ler, D.; Koprinska, I.; and Chawla, S. 2004b. A landmarker selection algorithm based on correlation and efficiency criteria. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, 296-306.

Ler, D.; Koprinska, I.; and Chawla, S. 2004c. A new landmarker generation algorithm based on correlativity. In *Proceedings of the International Conference on Machine Learning and Applications*, 178-185.

Michie, D.; Spiegelhalter, D.; and Taylor, C. 1994. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

Miller, A. 2002. *Subset Selection in Regression* (2nd Edition). Chapman and Hall/CRC.

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.

Vilalta, R.; and Drissi, Y. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2): 77-95.

Wickens, T. 1995. *The Geometry of Multivariate Statistics*. LEA Publishers.

Witten, I.; and Frank, E. 2000. Data Mining: Practical Machine Learning Tools with Java Implementations. Morgan Kaufmann.

Wolpert, D. 2001. The supervised learning no-free-lunch theorems. In Proceedings of the Soft Computing in Industry - Recent Applications, 25-42.