Fundamentals of Algorithm Selection for Classification

Daren Ler
School of Information Technologies, University of Sydney, NSW 2006, Australia
ler@it.usyd.edu.au

1. Introduction

What exactly is meant by classification? And in particular, what is meant by algorithm selection for classification? The first part of this paper addresses these questions and more specifically establishes the notation, definitions, descriptions and perspectives that comprise the fundamental concepts in algorithm selection for classification.

Subsequently, the second part of this paper is concerned with the measurement of algorithm performance, which intrinsically corresponds to general criteria that may be used to perform or to validate algorithm selection. Here, various methods of evaluating the generalisation performance of algorithms are described (e.g., cross-validation), as well as methods that may be used to compare and thus determine algorithm superiority.

2. Machine Learning

Machine learning is about the construction of computer programs that are able to learn. More specifically, a program is defined to learn from experience $\boldsymbol{E}$ when it is able to improve its performance over some learning problem $\boldsymbol{T}$, as measured by some performance criteria $\boldsymbol{P}$ (Mitchell, 1997). Correspondingly, in order to construct a program that learns, one essentially requires: (a) one or more well-specified learning tasks, (b) learning algorithms that hypothesise solutions for these tasks, and (c) performance metrics that grade the proficiency of any proposed hypothesis.

Typically, learning is required when a computer program must provide functionality in the presence of incomplete information, or rather, when it is either too expensive or too difficult[1] to develop a non-learning solution. Under such circumstances, the problem at hand is usually addressed as a learning problem for which a learning system must be derived. For example, in developing a program that filters spam email messages, deriving the part of the program that classifies an arbitrary email as spam or non-spam may be posed as a learning problem.

The design of a learning system is depicted in Figure 1, and essentially comprises three fundamental steps[2]: (i) selecting the type of training experience – i.e., how information regarding the learning problem is accessed, (ii) selecting an outline of the target function or concept, and the representation of its components – i.e., the specific form of the inputs and outputs, and conceptually, the intuitive mapping between them to be learned (accordingly, this

[1] Dietterich (2003) mentions that there are four general categories of problems that promote the need for machine learning: (i) problems for which there exist no human experts, (ii) problems where human experts exists, but where they are unable to explain their expertise, (iii) problems where phenomena are changing rapidly, and (iv) problems in which each computer user requires separate customisation.

[2] Note that the steps listed roughly follow the framework for designing a learning system defined in Mitchell (1997), but have been tailored to better fit the supervised learning paradigm, which this paper is primarily concerned with.

implies a performance metric to gauge learning success), and (iii) selecting a learning algorithm to hypothesise a specific form for the target function. Steps (i) and (ii) require knowledge regarding the problem domain in question, and correspond to Phase 1 in Figure 1, while Step (iii) requires knowledge of machine learning, and corresponds to Phases 2 through 5 in Figure 1. This entire process (typically) involves an iterative trial-and-error process that requires a considerable amount of expertise and computational effort, and it is likely that when designing a learning system, one might have to repeatedly revisit each step, each time refining the choices made. An in-depth description and analysis of this process is detailed in Brodley and Smyth (1997), and Kodratoff et al. (1992).

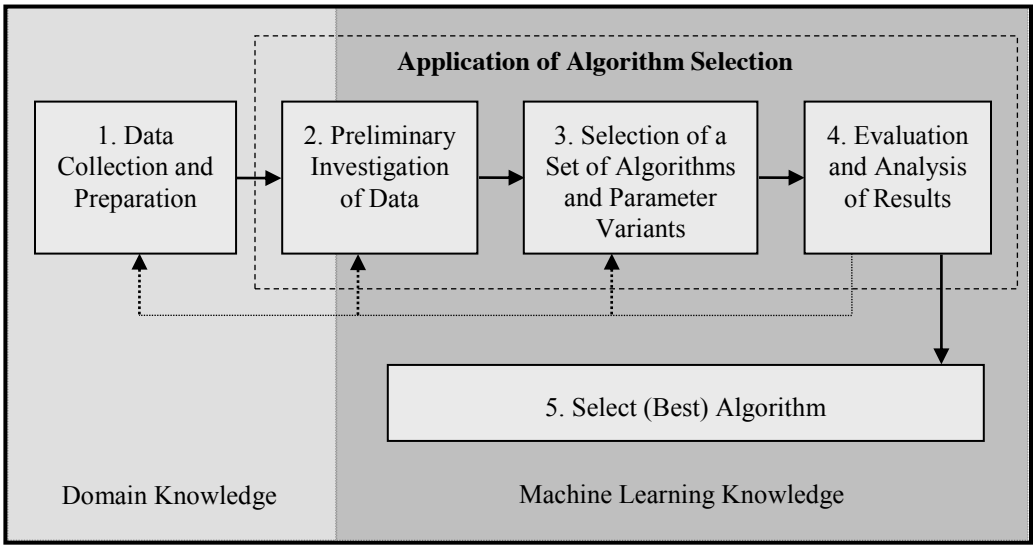

Figure 1. An overview of the iterative process of developing a learning system, and in particular, a classification solution.

Steps (i) and (ii) (and Phase 1 from Figure 1) essentially constitute a learning task[3] $\boldsymbol{t}$, which more specifically defines structure for $\boldsymbol{T}$, and correspondingly, defines the composition of $\boldsymbol{E}$. Consider the previous example, where the learning problem is how to identify if an arbitrary email message constitutes spam or non-spam. In this case, a corresponding learning task may be to take a sample of email messages and hypothesise a function that maps the characteristics of those emails (e.g., the frequency of a pre-defined list of words in the email subject and body, sender's email address, and so on) to the corresponding judgements regarding whether each of those emails is spam or non-spam. In this example, the training experience pertains to the given sample of emails, and the hypothesised form of the target function to $\boldsymbol{g}$: $\boldsymbol{X} \rightarrow \{0, 1\}$, where $\boldsymbol{g}$ is a function that maps the set of specified email characteristics $\boldsymbol{X}$ to a binary value indicating spam or non-spam.

In general, there are predominantly three paradigms of machine learning, or rather, three primary categories of learning tasks and algorithms: (a) supervised, (b) unsupervised, and (c) reinforcement learning. Supervised learning requires the discovery of patterns within a sample of data, where both the inputs (e.g., characteristics of emails) and the outputs (e.g., whether particular emails are spam or non-spam) are provided. Alternatively, in unsupervised learning, or clustering, output values are not provided, and patterns must be found by essentially

[3] Note that there is a distinction between a learning task and its associated learning problem; the learning problem is merely an intuitive description of what is to be learnt, whereas the learning task is a more specific and well-defined version of the learning problem. Accordingly, for each learning problem, several learning tasks may be defined.

grouping subsets of instances in the sample based on their similarity as defined by some similarity (i.e., distance) measure. Finally, reinforcement learning is concerned with learning from interaction with an environment, and from the consequences of action rather than from a sample of data. Literature on each of these paradigms is prolific; e.g., refer to Hastie et. al. (2001), Sutton and Barto (1998), Witten and Frank (2000), and Mitchell (1997).

This paper is primarily concerned with the third step of designing a learning system – i.e., selecting an appropriate algorithm for a given learning task $\boldsymbol{t}$. Furthermore, the work described only concentrates on learning tasks that correspond to classification problems, which are a proper subset of supervised learning problems.

More precisely, the general problem of algorithm selection for classification may be posed as follows. Given a learning task $\boldsymbol{t}$, which corresponds to a classification problem, and a set of applicable algorithms $\boldsymbol{A}$ that may be applied to solve $\boldsymbol{t}$, how then does one decide which specific algorithm $\boldsymbol{a} \in \boldsymbol{A}$ to utilise?

3. Supervised Learning and Classification

Supervised learning is perhaps the most well-studied branch of machine learning. In general, supervised learning is about how one can take a set of examples, and construct a hypothesis (e.g., function, or set of rules) that assumedly governs the mapping between the inputs and the outputs of any example from the same problem.

More specifically, a typical supervised learning scenario is as follows. Given a supervised learning task $\boldsymbol{t}$, which is represented by a dataset $\boldsymbol{s}$ comprising a set of n ordered pairs $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, the goal is to induce a hypothesis $\boldsymbol{h}$ that formulates a mapping(s) between the $\boldsymbol{x}$ values and y values. The hypothesis $\boldsymbol{h}$ assumedly encapsulates the information within the given dataset $\boldsymbol{s}$, and is typically used as a predictor to infers the value of y when presented with any instance of $\boldsymbol{x}$.

Each input or instance $\boldsymbol{x}_i$ is a vector in an m-dimensional space[4], $\boldsymbol{X} \subseteq \Re^m$, with each dimension in $\boldsymbol{X}$ corresponding to one attribute or feature of the dataset $\boldsymbol{s}$. Accordingly, $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,m})$, where each $x_{i,j}$ denotes the j-th attribute value of the i-th instance.

Each output y_i, also known as the label or class or target, is derived by applying the corresponding $\boldsymbol{x}_i$ to some unknown function $\boldsymbol{f}$ (i.e., $y_i = \boldsymbol{f}(\boldsymbol{x}_i)$), typically known as the target function or concept. (At times, it is also useful to perceive that $y_i = \boldsymbol{f}(\boldsymbol{x}_i) + \varepsilon_i$, where ε_i corresponds to noise component, which corrupts the true value of $\boldsymbol{f}(\boldsymbol{x}_i)$.)

If each y_i belongs to a range of continuous values (i.e., is quantitative), then the problem in question is a regression problem, whereas, if each y_i belongs to a set of discrete symbols (i.e., is qualitative), then the problem is a classification problem.

In a typical setting, each $(\boldsymbol{x}_i, y_i) \in \boldsymbol{s}$ is randomly drawn (or is assumed to be) from a fixed, although unknown probability distribution $\boldsymbol{D} = Pr[\boldsymbol{X}, \boldsymbol{y}]$ – i.e., the pairs in $\boldsymbol{s}$ are assumed to be independently and identically distributed.

The objective of a supervised learning algorithm $\boldsymbol{a}$ (e.g., the C4.5 decision tree algorithm (Quinlan, 1993)), is to use the given dataset $\boldsymbol{s}$ to generate a hypothesis $\boldsymbol{h}_a$ (which in the context of classification, is also often referred to as a classifier). Generally, algorithm $\boldsymbol{a}$ is said to be

[4] This m-dimensional space is conventionally referred to as the instance space.

trained using the dataset $\boldsymbol{s}$ (which in this context, may also be referred to as the training set), and thus is able learn a hypothesis $\boldsymbol{h_a}$.

Correspondingly, $\boldsymbol{h_a}$ equates to a mapping that generalises the relationship between the input (i.e., $\boldsymbol{x}$) and output (i.e. y) of each $(\boldsymbol{x}_i, y_i) \in \boldsymbol{s}$. Let $\boldsymbol{a}(\boldsymbol{s})$ denote the training process such that $\boldsymbol{a}(\boldsymbol{s}) \equiv \boldsymbol{h_a}$. Consequently, it is assumed[5] that $\boldsymbol{h_a}$ additionally generalises over any unseen instances drawn based on $\boldsymbol{D}$, and therefore is an approximation of the target function $\boldsymbol{f}$. More specifically, the algorithm $\boldsymbol{a}$ utilises the given dataset $\boldsymbol{s}$ to search a hypothesis space $\boldsymbol{H_a}(\boldsymbol{s})$ for a hypothesis $\boldsymbol{h_a} \in \boldsymbol{H_a}(\boldsymbol{s})$ to approximate the target function $\boldsymbol{f}$.

In summary, the goal of each learning algorithm is thus to find some $\boldsymbol{h_a}$, such that with high probability, an input $\boldsymbol{x}_{new} \in \boldsymbol{X}$ drawn according to $\boldsymbol{D}$ will be labelled correctly by $\boldsymbol{h_a}$.

To elaborate on this terminology, consider the following example. Suppose we are given a classification task $\boldsymbol{t}_{toy}$, which corresponds to a sample of 5 (noiseless) instances, where each instance consists of 4 Boolean-valued attributes and a Boolean-valued class. More specifically, in this case, each $\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4})$, where each $x_{i,j} \in \{0,1\}$, and correspondingly, each $y_i \in \{0,1\}$. Now, suppose that the target function in question is $\boldsymbol{f}(\boldsymbol{x}_i) = (x_{i,1} \wedge x_{i,2}) \oplus (x_{i,3} \vee x_{i,4})$, and that the instances in our dataset $\boldsymbol{s}_{toy}$ are drawn from a uniform distribution. Then a possible composition of the dataset $\boldsymbol{s}_{toy}$ is given in Table 1.

i	$\boldsymbol{x}_i$				y_i
	$x_{i,1}$	$x_{i,2}$	$x_{i,3}$	$x_{i,4}$	
1	0	0	1	0	1
2	0	0	1	1	1
3	0	0	1	1	1
4	1	0	0	0	0
5	1	1	0	0	1

Table 1: The dataset $\boldsymbol{s}_{toy}$., which consists of 5 instances, drawn via a uniform distribution over the instance space $\boldsymbol{X}_{toy}$, which is space consisting 4 Boolean attributes. The corresponding outputs y_i are given by the target function $\boldsymbol{f}(\boldsymbol{x}_i) = (x_{i,1} \wedge x_{i,2}) \oplus (x_{i,3} \vee x_{i,4})$.

Consider an algorithm ***a-simple***, which considers functions that correspond to any combination of disjunctions over the given attributes (e.g., $x_{i,1} \vee \ldots \vee x_{i,m}$), and then randomly outputs (as its hypothesis) any one of these functions that is consistent with the given dataset. Upon applying $\boldsymbol{s}_{toy}$ to ***a-simple***, the resultant hypothesis space $\boldsymbol{H}_{\boldsymbol{a\text{-}simple}}(\boldsymbol{s}_{toy})$, includes the hypotheses listed in Table 2. Among these, only 2 are consistent with the dataset $\boldsymbol{s}_{toy}$: $\boldsymbol{h}_4 = x_{i,2} \vee x_{i,3}$, and $\boldsymbol{h}_{10} = x_{i,2} \vee x_{i,3} \vee x_{i,4}$. Suppose that in this case, the hypothesis that is output is $\boldsymbol{h}_4$ – i.e., $\boldsymbol{a\text{-}simple}(\boldsymbol{s}_{toy}) = \boldsymbol{h}_4$. Unfortunately however, while $\boldsymbol{h}_4$ is consistent with the examples in $\boldsymbol{s}_{toy}$, is it distinctly different from $\boldsymbol{f}$. Using $\boldsymbol{h}_4$ as a solution (i.e. estimator) for $\boldsymbol{f}$ will probably prove to be a very bad idea.

Notice that it is thus the hypothesis space $\boldsymbol{H_a}(\boldsymbol{s})$ and the output hypothesis $\boldsymbol{h_a}$ that ultimately determines the utility of the algorithm $\boldsymbol{a}$ over that dataset $\boldsymbol{s}$, and more importantly, over the learning task $\boldsymbol{t}$.

Returning to the problem of algorithm selection, it thus becomes clear that selecting an algorithm given some $\boldsymbol{s}$ equates to finding some $\boldsymbol{a}$ that firstly, considers a hypothesis space

[5] Mitchell (1997) refers to this as the inductive learning hypothesis, which states that a hypothesis that approximates the target function $\boldsymbol{f}$ well over a sufficiently large set of instances will also approximate $\boldsymbol{f}$ well over unobserved instances.

$\boldsymbol{H_a(s)}$ containing a hypothesis $\boldsymbol{h^*}$ that is acceptably close to, or matches $\boldsymbol{f}$, and secondly, that is able to isolate and output $\boldsymbol{h^*}$. More precisely, algorithm selection for some given classification dataset $\boldsymbol{s}$, corresponds to the search for some algorithm $\boldsymbol{a}$ such that $\boldsymbol{h_i \approx f}$, where $\boldsymbol{h_i \in H_a(s)}$ and $\boldsymbol{a(s) = h_i}$.

j	$\boldsymbol{h_j}$
1	$x_{i,1} \vee x_{i,2}$
2	$x_{i,1} \vee x_{i,3}$
3	$x_{i,1} \vee x_{i,4}$
4	$x_{i,2} \vee x_{i,3}$
5	$x_{i,2} \vee x_{i,4}$
6	$x_{i,3} \vee x_{i,4}$
7	$x_{i,1} \vee x_{i,2} \vee x_{i,3}$
8	$x_{i,1} \vee x_{i,2} \vee x_{i,4}$
9	$x_{i,1} \vee x_{i,3} \vee x_{i,4}$
10	$x_{i,2} \vee x_{i,3} \vee x_{i,4}$
11	$x_{i,1} \vee x_{i,2} \vee x_{i,3} \vee x_{i,4}$

Table 2: The composition of the hypothesis space $\boldsymbol{H_{a\text{-}simple}(s_{toy})}$. Note that ***a-simple*** is an algorithm that considers functions corresponding to any combination of disjunctions over the given attributes. It randomly outputs any of these functions consistent with the given dataset. The dataset $\boldsymbol{s_{toy}}$ is described in Table 1.

4. Generalisation and Inductive Bias

Supervised learning, and consequently, learning for classification problems, inherently implies inductive[6] learning.

The only information available to a supervised learning algorithm $\boldsymbol{a}$, is typically[7] some dataset of labelled instances $\boldsymbol{s}$. However, the dataset $\boldsymbol{s}$ does not explicitly provide any information about the label y_{new} of any unseen instance $\boldsymbol{x}_{new}$ (i.e., where $(\boldsymbol{x}_{new}, y_{new}) \notin \boldsymbol{s}$). Analogously, any algorithm $\boldsymbol{a}$, that is employed to perform supervised learning will face the problem of (inductive) generalisation – i.e., after analysing only a (typically small) sample of labelled instances, the algorithm $\boldsymbol{a}$ will be required to output a hypothesis $\boldsymbol{h_a}$ that matches the target function $\boldsymbol{f}$ over any arbitrary example from the same problem.

Accordingly, the process of generalisation for any given supervised learning algorithm $\boldsymbol{a}$ essentially relates to the definition of some hypothesis space $\boldsymbol{H_a(s)}$, and correspondingly, to the utilisation of some mechanism to search $\boldsymbol{H_a(s)}$ for a hypothesis $\boldsymbol{h_a}$. However, over which $\boldsymbol{H_a(s)}$ should the search conducted? And how should the search itself be conducted? If the decisions regarding these two questions are inadequate, the generated hypothesis might be tragically different from the target function in question, in which case, we would have failed in our mission to generalise – e.g., as was the case in the ***a-simple*** and $\boldsymbol{s_{toy}}$ example from Section 3.

4.1 The Need for Bias

The hypothesis space an algorithm searches essentially corresponds to its hypothesis representation, or rather, to the structure that is imposed by that algorithm over potential

[6] Induction can generally be described as the procedure of developing general explanatory hypotheses to account for a set of facts – e.g., concluding that all planetary orbits are parabolic – after only actually examining a few. In its broadest sense, it involves reaching conclusions about the unobserved on the basis of what is observed.
[7] In some cases, other background knowledge regarding the learning problem is also available.

hypotheses. For example, the algorithm ***a-simple*** from Section 3 defines possible hypotheses as functions corresponding to any combination of disjunctions over the given (discrete-valued) attributes. Thus, given some classification dataset $\boldsymbol{s}$, the hypothesis space defined by ***a-simple***, $\boldsymbol{H}_{\textit{a-simple}}(\boldsymbol{s})$, will only include hypotheses consistent with its hypothesis representation – i.e., hypotheses that correspond to disjunctive combinations of discrete-valued attributes.

The generalisation success of any supervised learning endeavour hinges on whether a hypothesis $\boldsymbol{h}^*$, which matches, or at least closely approximates $\boldsymbol{f}$, can be found. Accordingly, a precondition for generalisation success is that the hypothesis representation imposed by the chosen learning algorithm is able to represent $\boldsymbol{h}^*$, or analogously, that $\boldsymbol{h}^*$ is contained within the resultant hypothesis space.

One obvious way to ensure that $\boldsymbol{h}^*$ is present in the hypothesis space is to consider a hypothesis space capable of representing every possible version of the target function $\boldsymbol{f}$.

Given a learning task $\boldsymbol{t}$, let the concept space $\boldsymbol{C}$ denote the set of all possible target functions applicable to $\boldsymbol{t}$ a priori (i.e., prior to the exposure to any instances and their labels). For example, consider the learning task in the ***a-simple*** and $\boldsymbol{s}_{toy}$ example from Section 3. Here, prior to training, we know that there are 4 Boolean-valued attributes, and further, that the class is Boolean-valued. Correspondingly, the instance space $\boldsymbol{X}$ for this problem contains 2^4 unique instances; these are listed in Table 3. The concept space in question thus corresponds to the set of all possible 16 bit binary strings – since each corresponds to one possible version of labelling over $\boldsymbol{X}$. Thus, in this particular example, the concept space contains 2^{16} possible target functions. (Notice that a priori, any of the 2^{16} 16 bit strings is equally valid since we have not as yet seen any evidence to the contrary.)

i	$\boldsymbol{x}_i$				y_i
	$x_{i,1}$	$x_{i,2}$	$x_{i,3}$	$x_{i,4}$	
1	0	0	0	0	0
2	0	0	0	1	1
3	0	0	1	0	1
4	0	0	1	1	1
5	0	1	0	0	0
6	0	1	0	1	1
7	0	1	1	0	1
8	0	1	1	1	1
9	1	0	0	0	0
10	1	0	0	1	1
11	1	0	1	0	1
12	1	0	1	1	1
13	1	1	0	0	1
14	1	1	0	1	0
15	1	1	1	0	0
16	1	1	1	1	0

Table 3: The truth table for the dataset $\boldsymbol{s}_{toy}$ (which is described in detail in Section 3), where $y_i = \boldsymbol{f}(\boldsymbol{x}_i) = (x_{i,1} \wedge x_{i,2}) \oplus (x_{i,3} \vee x_{i,4})$.

For each labelled instance exposed to (that is devoid of noise), the number of valid target functions is reduced as the label assigned to the instance in question invalidates any potential target function that infers otherwise.

Let the size of an instance space $\boldsymbol{X}$, or rather, the number of possible unique instances that could inhabit $\boldsymbol{X}$, be denoted $|\boldsymbol{X}|$, and the size of the concept space, or rather, the number of possible target functions that could be defined over $\boldsymbol{X}$ a priori, be denoted $|\boldsymbol{C}|$. Then, given c possible labels, $|\boldsymbol{C}| = c^{|\boldsymbol{X}|}$. When presented with k labelled instances, which are assumed to be free of classification noise, the resultant set of valid target functions (i.e., the target functions consistent with the k labelled instances) is thus $c^{|\boldsymbol{X}| - k}$. Any of these $c^{|\boldsymbol{X}| - k}$ target functions will thus be equally valid for classifying an instance in $\boldsymbol{X}$.

Typically, only a small subset of the instance space will be provided for learning. This means that when considering a hypothesis space equivalent to the concept space, the set of hypotheses consistent with the data (i.e., the set of valid target functions) will be large, and probably unmanageably large. Furthermore, when the learning task in question includes real-valued attributes, then the possible number of unique instances (prior to some form of discretisation[8]) may be infinite, meaning that there may never be enough labelled instances to sufficiently shrink the hypothesis space. If one persists in utilising the concept space as a hypothesis space, some means to discriminate between the (many) equally eligible hypotheses will be required.

Returning to the ***a-simple*** and $\boldsymbol{s}_{toy}$ example, we note that with exposure to the 5 instances in Table 1, the number of valid target functions is reduced to $2^{16 - 5}$. Consequently, when considering the label for a previously unseen instance, for example, $\boldsymbol{x}_{new} = (0, 0, 0, 0)$, there is no evidence to suggest any preference over any of the 2048 valid target functions. On the other hand, when given the same 5 instances, the hypothesis space of the ***a-simple*** algorithm only contains 2 hypotheses that are consistent. Ultimately, the ***a-simple*** algorithm was only able to output a hypothesis (albeit an incorrect one) because it imposed assumptions regarding the structure of potential target function and about the likelihood of each consistent hypothesis conforming to this structure.

Essentially, any algorithm that must generalise must make assumptions about the learning task. An unbiased algorithm, which makes no a priori assumptions regarding the identity of the target function, has no rational basis for classifying unseen instances (Mitchell, 1990, 1997).

To generalise, bias[9] is required.

4.2 Inductive Bias

The a priori assumptions[10] that are embedded in a supervised learning algorithm are typically referred to as the inductive bias of that algorithm. Essentially, an inductive bias pertains to the policy by which an algorithm generalises beyond the observed data in order to infer the labels of unseen instances (Mitchell, 1997; DesJardins and Gordon, 1995).

---

[8] Discretisation is concerned with the process of transferring continuous models and equations into discrete counterparts. In this case, it pertains to the process of converting continuous (real-valued) attributes into relevant discrete attributes. This is typically done via some form of thresholding.

[9] The term bias has a different (though not totally unrelated) meaning in statistics. However, given the close relationship between statistics and machine learning, it should be made clear that the term bias here refers to inductive bias (i.e., the set of assumptions inherent in a learning algorithm), which is the subject of Section 4.2; it does not refer to statistical bias. For a more detailed description of the distinction between inductive bias and statistical bias, refer to (Dietterich and Kong, 1995).

[10] Intuitively speaking, these assumptions typically correspond to the form of correlation that is assumed to exist between the inputs $\boldsymbol{X}$ and outputs $\boldsymbol{y}$ of the given learning task (and its corresponding dataset).

More precisely, Mitchell (1997) defines the inductive bias of an inductive learning algorithm $\boldsymbol{a}$ as any minimal set of assertions $\boldsymbol{B}$, such that for any (labelled) dataset $\boldsymbol{s}$ (whose instances are drawn from $\boldsymbol{X}$) and corresponding target function $\boldsymbol{f}$, the classification of any instance $\boldsymbol{x}_i \in \boldsymbol{X}$ follows deductively from $\boldsymbol{B}$, $\boldsymbol{s}$ and $\boldsymbol{x}_i$. In other words, by conceptualising the set of assumptions $\boldsymbol{B}$ used to make an inductive inference $\boldsymbol{h}_a(\boldsymbol{x}_i)$ (which is justified by $\boldsymbol{s}$ and $\boldsymbol{x}_i$), a deductive[11] justification for the same inference may be made (from $\boldsymbol{B}$, $\boldsymbol{s}$ and $\boldsymbol{x}_i$).

Additionally, since the inductive bias of an algorithm reflects the form of the hypothesis space it considers, as well as the mechanism with which it searches that hypothesis space, it may thus be segmented into two different forms of bias: representation bias, and preference bias (Utgoff, 1986; Rendall, 1986; Dietterich, 1990; Mitchell, 1997).

Representation bias (also known as restriction or language bias (Mitchell, 1997)) corresponds to assumptions regarding the hypothesis representation, or rather, to the structure that an algorithm imposes over hypothesis space. Alternatively, preference bias (also known as procedural bias (DesJardins and Gordon, 1995)) corresponds to the manner in which an algorithm searches its defined hypothesis space, or rather, to the preferences it specifies over the hypotheses within its defined hypothesis space.

Successful generalisation therefore depends on the appropriateness of the inductive bias (i.e., the representation bias and preference bias) that is adopted to solve a learning task. Essentially, when the inductive bias, or rather, the assumptions $\boldsymbol{B}$ are correct (and if $\boldsymbol{s}$ is noise-free), the classifications made by corresponding algorithm $\boldsymbol{a}$ will also be correct. Conversely, the classifications will be incorrect if $\boldsymbol{B}$ is incorrect.

Thus, given a learning task $\boldsymbol{t}$, and an accompanying classification dataset $\boldsymbol{s}$, the task of algorithm selection over a set of applicable algorithms $\boldsymbol{A}$ equates to the selection of some $\boldsymbol{a} \in \boldsymbol{A}$ with the right assumptions $\boldsymbol{B}$, or rather, the right inductive bias relative to $\boldsymbol{t}$. Essentially, the more attuned the inductive bias $\boldsymbol{B}$ is to the underlying structure inherent in $\boldsymbol{t}$, the better the approximation $\boldsymbol{h}_i \approx \boldsymbol{f}$ is expect to be (where $\boldsymbol{h}_i \in \boldsymbol{H}_a(\boldsymbol{s})$ and $\boldsymbol{a}(\boldsymbol{s}) = \boldsymbol{h}_i$).

5. Evaluating Algorithm Performance

Determining the applicability of a supervised learning algorithm (and accordingly, the applicability of its associated inductive bias) implies a mechanism to gauge performance. Thus, fundamental to supervised learning, and in particular, to algorithm selection, is the ability to evaluate the performance of algorithms.

In supervised learning, it is necessary to evaluate algorithms so as to gauge the viability of any hypothesis that is generated, and thus, whether that hypothesis may be applied or if a new one should be sought. Additionally, such evaluation may comprise an integral part of a learning algorithm or system (e.g., in the post-pruning of decision trees (Quinlan, 1993), to derive hypothesis weights in the AdaBoost algorithm (Freund and Schapire, 1997), and in wrapper-based feature selection (Guyon and Elisseeff, 2003)).

In terms of algorithm selection, the evaluation of algorithm performance is necessary for reasons similar to those mentioned above. Essentially, the manner in which we evaluate algorithms provides the metrics with which to compare, and thus, to choose between them.

The selection of an algorithm is dependent upon its evaluated performance.

[11] To clarify, deduction is the process of drawing out (i.e., making explicit) the implications of one or more premises. If one infers correctly what the premises imply, then the inference (i.e., conclusion) is said to be valid.

Therefore, given some classification dataset $\boldsymbol{s}$, the selection of an algorithm $\boldsymbol{a} \in \boldsymbol{A}$ (with the most appropriate inductive bias $\boldsymbol{B}$, and therefore the best approximation for $\boldsymbol{h}_i \approx \boldsymbol{f}$, given $\boldsymbol{h}_i \in \boldsymbol{H}_{\boldsymbol{a}}(\boldsymbol{s})$ and $\boldsymbol{a}(\boldsymbol{s}) = \boldsymbol{h}_i$) requires some performance metric $\boldsymbol{P}$. Essentially, $\boldsymbol{P}$ affords a means of comparing or ranking the various algorithm options in $\boldsymbol{A}$, thus establishing a criteria for selection.

5.1 Measuring Generalisation Ability

Depending on the goals of the learning problem, different measures of performance may be adopted. However, the evaluation of supervised learning algorithms is typically (and primarily) concerned with generalisation ability, or in other words, with how well the hypothesis generated via an algorithm is able to generalise beyond the given set of labelled instances – i.e., how closely some $\boldsymbol{h}_i \approx \boldsymbol{f}$ (Wolpert, 2001).

Consequently, this paper focuses solely on generalisation performance as an algorithm selection criterion. More specifically, the performance metric $\boldsymbol{P}$ that this paper is concerned with includes only those metrics which gauge generalisation ability.

Other performance metrics include efficiency, simplicity, usability, comprehensibility, and various other forms of accuracy or error (e.g., resubstitution error[12]). For a discussion of performance metrics beyond accuracy, refer to Giraud-Carrier (1998). And for multi-criteria performance metrics, refer to Nakhaeizadeh and Schnabel (1997), Keller et al. (2000) and Soares et al. (2000).

Dietterich (1997) states that the generalisation ability of learning algorithms (i.e., the generalisation ability of the hypotheses that they output) are determined by how well their inductive biases match the true structure of the learning problem, and thus, that "fundamental research in machine learning is inherently empirical". The primary argument stems from the results of no free lunch (NFL) theorems (Wolpert, 1996b, 1996a, 2001, 2000) – and analogously, on the similar albeit more simplified conservation law for generalisation performance (Schaffer, 1994), which (loosely speaking) show that if all unknown target functions $\boldsymbol{f}$ are equally likely, all learning algorithms will have identical generalisation ability, regardless of the inductive bias that each adopts (this includes an algorithm that adopts random guessing)[13]. In essence, these theorems follow from the observation that a given sample of labelled instances $\boldsymbol{s}$ provides no information regarding the label y_{new} of an instance $\boldsymbol{x}_{new}$, where $(\boldsymbol{x}_{new}, y_{new}) \notin \boldsymbol{s}$. As a consequence, there is generally no mathematical basis for inferring $\boldsymbol{f}(\boldsymbol{x}_{new})$ solely based on $\boldsymbol{s}$.

Accordingly, most methods of evaluating generalisation ability involve experimentation over the given dataset. Typically, this involves some variation or extension of the standard holdout procedure, where a subset ***s-train*** (known as the training set) is sampled from the given dataset

[12] Resubstitution error corresponds to the proportion of misclassifications a hypothesis makes over the set of data that was used to train it.

[13] From a different perspective, the NFL theorems preclude the existence of a generic mechanism to evaluate the generalisation ability of learning algorithms (Wolpert, D. Personal communication (via email), 2006). However, this is only assuming that all target functions are equally likely. The typical argument posed in defiance of the NFL theorems is that a uniform probability distribution over all target functions is unlikely in the real world (Giraud-Carrier and Provost, 2005; Rao et al., 1995), and that in general, machine learning practitioners do not apply the principal of indifference (Knaela, 1949) when attempting to solve a given learning problem (i.e., they assume some target functions are more likely than others). Giraud-Carrier and Provost (2005) argue that the fundamental belief in machine learning is that the process that presents us with learning problems induces a non-uniform probability distribution over the target functions that is explicitly or implicitly known (at least to a useful approximation). Unfortunately, this claim cannot be tested – i.e., no experimental data can confirm it (as established by the NFL theorems) (Wolpert, D. Personal communication (via email), 2006). Essentially, the NFL theorems relate back to Hume's (1740, 1748) famous conclusion that induction has no rational basis. Fundamentally, induction cannot be evaluated without the help of induction - refer to Bensusan (1999) for a discussion of this topic.

s, and utilised to train[14] the algorithm in question, while the complement subset ***s-test*** (known as the test set – i.e., where ***s-test*** = ***s*** \ ***s-train***) is used to test the hypothesis generated via ***s-train***. (In the typical holdout setting, ***s-train*** comprises 2/3 of ***s***, while ***s-test*** comprises the remaining 1/3.)

The general idea is that by evaluating the learned hypothesis using a sample that is independent of the process that generated it, the resultant measurement will not be optimistically biased, and will thus provide a more accurate (i.e., a statistically unbiased) estimation of its generalisation ability (Mitchell, 1997; Witten and Frank, 2000).

In the general case, various loss functions[15] may be applied when evaluating generalisation ability (typically dependent on the type of supervised learning problem). However, as this paper is primarily concerned with classification problems, the specific measurement that will be used is the misclassification or error rate of the hypothesis (i.e., 0-1 loss), which corresponds to the proportion of instances from the given test set that the hypothesis in question classifies incorrectly.

5.2 Confidence Intervals for the Error Rate of a Hypothesis

When measuring the error rate of a learned hypothesis ***h***, the objective is ultimately to discover the generalisation ability of ***h*** over the entire (unknown) distribution ***D*** of instances. Thus, is it natural to wonder how closely the error rate measured on a test set (i.e., ***s-test***) will approximate the true error rate (i.e., how closely it will approximate the probability that ***h*** will misclassify a single randomly drawn instance from the distribution ***D***).

Let $error_{s\text{-}test}(\boldsymbol{h})$ denote the proportion of instances in the sub-sample ***s-test*** that ***h*** misclassifies, and correspondingly, let $error_{D}(\boldsymbol{h})$ denote the probability that ***h*** will misclassify a single randomly drawn instance from the distribution ***D***. Via statistical reasoning, one may compute the confidence interval for $error_{D}(\boldsymbol{h})$ based upon $error_{s\text{-}test}(\boldsymbol{h})$. More specifically, given that ***s-test*** contains n instances (drawn independent of one another, and of ***h***, according to ***D***), and where n is large, with $100(1 - \alpha)\%$ probability (where $0 \leq \alpha \leq 1$),

$$error_{D}(\boldsymbol{h}) \approx error_{s\text{-}test}(\boldsymbol{h}) \pm z_{\alpha/2} \sqrt{\frac{error_{s\text{-}test}(\boldsymbol{h})(1 - error_{s\text{-}test}(\boldsymbol{h}))}{n}}$$

where the constant $z_{\alpha/2}$ is chosen depending on the desired confidence level, which are given in Table 4.

$100(1 - \alpha)\%$:	50.0%	75.0%	90.0%	95.0%	99.0%	99.5%	99.9%
Constant $z_{\alpha/2}$:	0.67	1.15	1.64	1.96	2.58	2.81	3.30

Table 4: The standard values $z_{\alpha/2}$ extending from the centre of the standard normal curve (i.e., $z = 0$) that correspond to $100(1 - \alpha)\%$ of the area under the standard normal curve about its centre. Accordingly, $100(1 - \alpha)\%$ reflects the probability that $error_{D}(\boldsymbol{h})$ is within $z_{\alpha/2}$ standard deviations in either direction of $error_{s\text{-}test}(\boldsymbol{h})$.

It is important to note that this confidence interval only corresponds to error rates, and that it is only an approximate confidence interval. Essentially, the primary reason why the interval is only an approximate is because it utilises the normal approximation for a binomial distribution.

[14] Note that if the training process additionally requires some form of evaluation (e.g., for parameter tuning), then it is common to form a validation set ***s-validation*** that is independent of both ***s-train*** and ***s-test***.

[15] A loss function is a function that maps an event (i.e., an element of a sample space) onto a real number representing the cost or regret associated with the event.

Regarding this issue, various rules of thumb have been suggested; Witten and Frank (2000) suggest $n > 100$, while Mitchell (1997) suggests that $n(error_{s\text{-}test}(\boldsymbol{h})(1 - error_{s\text{-}test}(\boldsymbol{h}))) \geq 5$, and both Weiss (1993) and Devroye (1986) suggest $n(error_{s\text{-}test}(\boldsymbol{h})) > 5$ and $n(1 - error_{s\text{-}test}(\boldsymbol{h})) > 5$. In general, the larger n is, and the less the skew of the underlying binomial distribution, the better the approximation will tend to be.

For a more detailed description of how the confidence interval for $error_D(\boldsymbol{h})$ is formulated, refer to Appendix A.

More recently, Langford (2005b) has advocated the use of more precise confidence intervals in the form of the "test set bound", which directly utilises the binomial distribution instead of its normal approximation. The motivation for this bound is based upon our general desire to derive hypotheses that have error rates that are close to 0, and on the fact that the binomial distribution is not similar to a normal distribution when the error rate is near 0. Consequently, adopting the normal approximation for the construction of error rate confidence intervals leads to fundamentally misleading results that are both pessimistic and (much worse) optimistic. Conversely, the "test set bound" approach is never optimistic, and derives an error rate confidence interval that more appropriately has an upper and lower bound in [0,1].

5.3 Cross-validation and Other Evaluation Methods

Thus far, evaluating the generalisation ability of an algorithm $\boldsymbol{a}$ has been associated with evaluating the error rate of a single hypothesis $\boldsymbol{h}_a$, where $\boldsymbol{a}(\boldsymbol{s\text{-}train}) = \boldsymbol{h}_a$. However, data is typically very limited in practice, and consequently, determining the generalisation ability of $\boldsymbol{a}$ is made increasing difficult due to the potential sources of variation[16] present in evaluation.

On reflection, one notices that while the standard holdout procedure is a simple and computationally inexpensive, it clearly makes inefficient use of the given data; only 2/3 of the data is utilised for training, since the remaining 1/3 is required for testing. Essentially, the more instances utilised for testing, the smaller the training set will be, and the higher the chance that the generated hypotheses will vary. Conversely, fewer test instances will increase the variance over measured error rates, and accordingly, result in wider confidence intervals (notice from Section 5.2 that the size of the confidence interval for $error_D(\boldsymbol{h})$ is inversely related to n). In general, when the given dataset is large, both the test and training sets tend to be appropriately large, and standard holdout testing will tend to provide a reliable estimate (Mitchell, 1997; Witten and Frank, 2000). However, it should be avoided when data is not plentiful.

Consequently, to limit variation in evaluation, and thus, to glean a less sketchy account of the generalisation ability of $\boldsymbol{a}$, several alternative methods have been developed that tend to more efficiently and effectively utilise the given dataset $\boldsymbol{s}$. Such methods typically involve the generation of multiple hypotheses using different $\boldsymbol{s\text{-}train}$ and $\boldsymbol{s\text{-}test}$ splits. The results over the various splits are then aggregated in some fashion to report the viability of the algorithm in question. Two such methods are cross-validation (Efron, 1979; Stone, 1974, 1977) and bootstrapping (Efron and Tibshirani, 1993; Efron, 1983).

In k-fold cross-validation, the given dataset $\boldsymbol{s}$ is divided into k mutually exclusive subsets (i.e., the folds) $\boldsymbol{fold}_1, \ldots, \boldsymbol{fold}_k$ of approximately equal size. More specifically, given that $|\boldsymbol{s}| = n$, let $m = \lfloor n/k \rfloor$, $p = n \bmod k$ (note additionally, $p = n - mk$), and $q = k - i$; there are thus p folds of $m + 1$ instances, and q folds of m instances. For each $\boldsymbol{fold}_i$, the algorithm $\boldsymbol{a}$, whose performance is being measured is then trained using $\boldsymbol{s} \setminus \boldsymbol{fold}_i$, and tested via $\boldsymbol{fold}_i$. (Notice that each instance

[16] Dietterich (1998) identifies four sources of variation that may cause erroneous approximations of generalisation ability (and in this case, error rate): (i) test set variation, (ii) training set variation, (iii) hypothesis variation due to the non-deterministic nature of the learning algorithm in question, and (iv) random classification error.

in $\boldsymbol{s}$ is tested exactly once.) The cross-validated performance measurement (e.g., error rate) is then taken as the average over the k folds. Accordingly, the error rate of $\boldsymbol{a}$ for $\boldsymbol{s}$, measured using cross-validated, is:

$$\textit{error-CVs}(\boldsymbol{a}) \equiv \frac{1}{k}\sum_{i=1}^{k} \textit{error}_{\textbf{\textit{fold}}_i}(\boldsymbol{a}(\boldsymbol{s} \setminus \textbf{\textit{fold}}_i))$$

(Notice that the hypothesis generated over each of the k iterations is different since training is done with a different subset of $\boldsymbol{s}$.)

Given n and k, the number of possible k-fold splits is:

$$\left[\prod_{i=1}^{p}\binom{n-(m+1)(i-1)}{m+1}\right]\times\left[\prod_{j=1}^{q}\binom{n-(m+1)p-m(j-1)}{m}\right]$$

Consequently, the opportunity to vary between different k-fold subsets may still be high. One way to restrict the number of possible splits is to use a method called stratification.

Stratified k-fold cross-validation works like regular k-fold cross-validation, with one exception; each fold generated must roughly[17] maintain the same class label distribution as that observed in the given dataset $\boldsymbol{s}$. Essentially, by adding this condition, the number of possible splits is reduced, and the chance that certain classes are left out of any training or testing set is lessened (which may improve the accuracy of the error rate estimate).

A special case of k-fold cross-validation is leave-one-out (or n-fold) cross-validation (Lachenbruch and Mickey, 1968). Here, since $k = n$, n mutually exclusive subsets must be found, meaning that each fold must hold exactly one instance. Consequently, the leave-one-out cross-validation method is deterministic, and utilises the maximal amount of instances in training (given that a non-empty test set is required). However, this method of evaluation is computationally expensive; with n folds, training (each of) the learning algorithm(s) in question must be performed n times (over $n - 1$ instances). Furthermore, the individual test sets each only hold a single instance, and thus, cannot follow the class distribution over $\boldsymbol{s}$ (unless $\boldsymbol{s}$ is trivial and only contains a single possible class label).

Alternatively, in bootstrapping (and specifically in this case, the 0.632 bootstrap), a bootstrap sample ***s-boot*** is created by randomly sampling (with replacement) n instances from the given dataset $\boldsymbol{s}$ (where $|\boldsymbol{s}| = n$). As the bootstrap sample is sampled with replacement, the probability that an instance in $\boldsymbol{s}$ is not chosen is $(1 - 1/n)^n$, or approximately, $e^{-1} \approx 0.368$. Accordingly, ***s-boot*** is utilised as the training set, while the instances that were not chosen when sampling with replacement (i.e., $\boldsymbol{s} \setminus$ ***s-boot***) are utilised in the test set.

As with standard holdout testing, bootstrapping only utilises a fraction (roughly 63.2%) of the given data for training, resulting in a pessimistic estimation of performance. To compensate for this, part of the performance over the training set is combined with the performance of the test set. More specifically, the bootstrap error rate estimate is given by $\textit{error-boot}_{\boldsymbol{s}}(\boldsymbol{h}) = 0.632(\textit{error}_{\textbf{\textit{s-boot}}}(\boldsymbol{h})) + 0.368(\textit{error}_{\textbf{\textit{s-test}}}(\boldsymbol{h}))$, where $\boldsymbol{h} = \boldsymbol{a}($***s-boot***$)$, and ***s-test*** $= \boldsymbol{s} \setminus$ ***s-boot***. This process is then repeated k times (Efron (1983) recommends $k = 200$), and the results averaged.

5.4 Determining an Evaluation Method to Adopt

Given some classification dataset $\boldsymbol{s}$, the selection of an algorithm $\boldsymbol{a}_i \in \boldsymbol{A}$ may thus (simplistically) be performed by first evaluating the error rate of each $\boldsymbol{a}_i \in \boldsymbol{A}$ on $\boldsymbol{s}$ (employing

[17] For a given class label y_i, the number of instances in each fold with class y_i can at most differ by 1.

one of the methods described in Section 5.3 – e.g., 10×10-fold cross-validation), and then selecting the algorithm that attained the lowest error rate. However, which evaluation method should be utilised?

In general, several studies (e.g., Efron (1983), Bailey and Elkan (1993)) collaborate the conclusion that the leave-one-out cross-validation method produces almost (statistically) unbiased estimates of the true error, but with high variance, and conversely, that bootstrapping produces low variance, but high bias. Correspondingly, the general intuition about cross-validation is that the larger k grows, the lower the estimation bias, and the higher the variance (though stratification seems to help reduce this variance (Kohavi, 1995)).

Accordingly, various studies conclude in favour of regular and stratified 10-fold cross-validation (e.g., see Kohavi (1995), Weiss and Indurkyha (1994), Breiman and Spector, (1992)). In particular, Kohavi (1995) concludes that stratified 10-fold cross-validation is appropriate for algorithm selection, even if the computational power available is sufficient for more computationally intensive methods of evaluation. However, while Witten and Frank (2000) acknowledge that "the standard way of predicting the error rate of a learning technique given a single, fixed sample of data is to use stratified ten-fold cross-validation", they suggest that a more reliable error estimate can be obtained by averaging the error rates from 10×10-fold cross-validation – i.e., 10 different runs (or repetitions) of 10-fold cross-validation.

It should further be noted that the theoretical foundations (i.e., the gap between experimental observation and theoretical understanding) for the use of cross-validation to measure generalisation performance remains an open problem (Langford, 2005a). Langford suggests that due to the prevalence of cross-validation, the impact of such theory would be significant. However, the impact of such theory on any individual problem would be small due to only slightly improved judgement of success.

With regard to the NFL theorems (Wolpert, 1996b, 1996a, 2001, 2000; Schaffer, 1994), Wolpert suspects that the prior knowledge encapsulated in cross-validation is far more general and has a tighter fit to the real world (in the sense of inner product) than the prior knowledge traditionally incorporated into Bayesian analysis (Wolpert, D. Personal communication (via email), 2003). Accordingly, consistent with the (conventional) view that cross-validation is usually an accurate estimator of generalisation performance, Wolpert (1996a) suggests that cross-validation may have desirable head-to-head minimax behaviour[18]. However, he further suggests that a head-to-head minimax superior method may only be possible because the algorithms that people use are very similar to one another – i.e., almost all learning algorithms try to fit the data, but restrict the complexity of the hypothesis – and thus, it is "because people have had such limited imagination in constructing learning algorithms" that an algorithm (speculatively, cross-validation say) is "superior" to those currently known.

To the knowledge of the author, the general consensus of the machine learning community (despite the lack of theoretical validation), is that 10-fold or 10×10-fold cross-validation remains the default (and probably the most dependable – for its cost) method of evaluation.

6. Comparing the Performance of Multiple Algorithms

The discussion on evaluating performance has thus far been primarily focused on estimating the true error rate (i.e., generalisation performance) of an individual supervised learning algorithm. An intuitive method of performing algorithm selection (given some classification dataset $\boldsymbol{s}$) and

[18] The NFL theorems may be met by having comparatively many cases in which method $\boldsymbol{A}$ is just slightly worse than method $\boldsymbol{B}$, and comparatively few cases when method $\boldsymbol{A}$ is better than method $\boldsymbol{B}$ by a lot. In such a case, $\boldsymbol{A}$ is deemed to be head-to-head minimax superior to $\boldsymbol{B}$.

a set of candidate algorithms $\boldsymbol{A}$) would be to estimate the true error rate of each $\boldsymbol{a}_i \in \boldsymbol{A}$ and to select the algorithm with the lowest error rate. However, a direct comparison such as this may not be sufficient to significantly (i.e., in a statistical context) determine if an algorithm $\boldsymbol{a}_1$ really has a lower error rate than another algorithm $\boldsymbol{a}_2$, let alone discern if the differences in error rate among all the algorithms within $\boldsymbol{A}$ are statistically significant.

Essentially, to establish whether differences in estimated error rates reflect actual differences in true error, the correct statistical questions[19] must be posed, and consequently, the appropriate statistical tests conducted.

6.1 Comparing Two Algorithms

In many texts (e.g., Mitchell (1997)) the suggested approach for comparing the error rates of two algorithms, $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, is a paired test[20], or more specifically, a k-fold cross-validation paired t-test, which is generally performed as follows.

First, k folds $\boldsymbol{fold}_1$, …, $\boldsymbol{fold}_k$ are sampled from the given dataset $\boldsymbol{s}$ (as per k-fold cross-validation). For each $\boldsymbol{fold}_i$, $error_{\boldsymbol{fold}_i}(\boldsymbol{a}_1(\boldsymbol{s\text{-}train}))$ and $error_{\boldsymbol{fold}_i}(\boldsymbol{a}_2(\boldsymbol{s\text{-}train}))$ are measured, where $\boldsymbol{s\text{-}train} = \boldsymbol{s} \setminus \boldsymbol{fold}_i$, and $d_i = error_{\boldsymbol{fold}_i}(\boldsymbol{a}_1(\boldsymbol{s\text{-}train})) - error_{\boldsymbol{fold}_i}(\boldsymbol{a}_2(\boldsymbol{s\text{-}train}))$ is computed. Since each $error_{\boldsymbol{fold}_i}(\boldsymbol{a}_1(\boldsymbol{s\text{-}train}))$ and $error_{\boldsymbol{fold}_i}(\boldsymbol{a}_2(\boldsymbol{s\text{-}train}))$ is a mean (over the test instances in $\boldsymbol{fold}_i$), the central limit theorem[21] applies (assuming that the number of instances in $\boldsymbol{fold}_i$ is large – i.e., assuming that roughly, $|\boldsymbol{fold}_i| \geq 30$), and consequently, the mean over the error rare differences $\bar{d}$ approximately follows a normal distribution.

A hypothesis test[22] may thus be conducted with the null hypothesis H_0: $\mu_d = 0$ and alternative hypothesis H_a: $\mu_d \neq 0$, in order to determine if $\boldsymbol{a}_1$ is significantly different from $\boldsymbol{a}_2$. More specifically, one can obtain the t-value for $\bar{d}$ (which approximately follows the t distribution[23] with $k - 1$ degrees of freedom):

$$t_{\bar{d}} = \frac{\bar{d}}{s_d / \sqrt{k}}, \text{where } s_d = \sqrt{\frac{1}{k-1}\sum_{i=1}^{k}(d_i - \bar{d})^2},$$

and the appropriate p-value[24] $p(\bar{d})$, and then conclude that both algorithms perform equally well if $-\alpha/2 < p(\bar{d}) < \alpha/2$, or else, that $\boldsymbol{a}_1$ is better than $\boldsymbol{a}_2$ if $p(\bar{d}) \geq \alpha/2$, or that $\boldsymbol{a}_2$ is better than $\boldsymbol{a}_1$ if $p(\bar{d}) \leq \alpha/2$ (given some significance value $0 \leq \alpha \leq 1$).

Unfortunately, while a paired t-test assumes that individual measurements (i.e., the difference between the error rates measured in each fold) are independent, this is not the case when k-fold

[19] For a taxonomy of the different statistical questions that arise in machine learning, refer to Dietterich (1998).
[20] Paired tests are tests where different hypotheses are evaluated over identical (test) samples.
[21] Given a set of independent and identically distributed random variables x_1, …, x_n governed by an arbitrary probability distribution with mean μ and finite variance σ^2. Then as $n \to \infty$, the central limit theorem states that the distribution governing $(\bar{x} - \mu)/(\sigma/\sqrt{n})$, where $\bar{x}$ is the mean over x_1, …, x_n approaches the standard normal distribution $N(0,1)$.
[22] A hypothesis test corresponds to the use of statistics to determine the probability that a given hypothesis is true. The usual hypothesis testing process consists of four steps: (1) formulating the null hypothesis H_0 and the alternative hypothesis H_a, (2) identifying a test statistic to assess the validity of H_0, (3) computation of the p-value (see point below), and (4) comparing the p-value to a significance value α; if $p \leq \alpha$, H_0 should be rejected in favour of H_a.
[23] The t-distribution (also referred to as Student's t-distribution) describes the distribution of a standardised (normally distributed) variable whose standard deviation is measured based on a sample estimation. More detailed descriptions, as well as t-value tables can be found in Neter et. al. (1988), Weiss (1993), and Weisstein (2001).
[24] The p-value of a hypothesis test is the probability of observing a value of the test statistic (which in this particular case is a t-value) that is at least as inconsistent with the null hypothesis H_0 as the value of the test statistic (i.e., t_i) actually observed. The smaller the p-value, the stronger the evidence against H_0. This translates to the area under the relevant t-distribution between $-t_i$ and t_i when H_a is $\mu_\delta \neq 0$ (i.e., test is two-tailed), or between $-\infty$ and t_i when H_a is $\mu_\delta < 0$ (i.e., test is left-tailed), or between t_i and $+\infty$ when H_a is $\mu_\delta > 0$ (i.e., test is right-tailed).

cross-validation is conducted. In k-fold cross-validation, $n(k-2)/(k-1)$ of the instances in any training set will overlap with another training set, meaning that the training set are not strictly independent. Consequently, this causes the variance to be understated and the Type I error[25] to be elevated (Dietterich, 1998).

As alternatives that overcome these problems, Dietterich (1998) recommends the use of the McNemar test (Everitt, 1977) (when running the algorithm(s) in question multiple times is inappropriate), or a 5×2-fold (i.e., 5-run 2-fold) cross-validation paired t-test. These tests are described in Appendix B.

More recently, Bouckaert and Frank (Bouckaert 2003a, 2003b, 2004, 2005; Bouckaert and Frank, 2004) propose the notion of replicability (of experiments), which refers to the probability that two runs of an experiment, on the same data set, with the same pair of algorithms and the same method of sampling the data, produces the same outcome. Based upon their various experiments, which included the comparison of many different sampling schemes and statistical tests (including the 5×2-fold cross-validation paired t-test (Bouckaert, 2003a; Bouckaert and Frank, 2004) and McNemar's test (Bouckaert, 2005)), they recommend using a sorted m×k-fold cross-validation paired t-test.

The sorted m×k-fold cross-validation procedure differs from conventional m×k-fold cross-validation in that the averaging is performed across the m i-th ranking fold error rates of the m runs instead of the conventional averaging across the k error rates of the k folds of each run.

Let $\boldsymbol{d}_i = \{d_{i,1}, d_{i,2}, \ldots, d_{i,k}\}$ denote the set of error rate differences (between the two algorithms in question) obtained over the k folds in i-th run, where each $d_{i\text{-}j}$ denotes the difference in error rate obtained over the j-th fold of the i-th run. (Note that each $d_{i\text{-}j}$ is computed in similar fashion to d_i from the k-fold cross-validation paired t-test described earlier in this section.) Also, let $rank_l(\boldsymbol{d}_i)$ denote the l-th ranked (i.e., l-th highest) difference among the differences in $\boldsymbol{d}_i$, and let the mean over $rank_l(\boldsymbol{d}_1), \ldots, rank_l(\boldsymbol{d}_m)$ (for $l = 1, \ldots, k$) be denoted δ_l, where:

$$\delta_l = \frac{1}{m}\sum_{i=1}^{m} rank_l(\boldsymbol{d}_i)$$

The sorted m×k-fold cross-validation paired t-test, is thus a test utilising the statistic:

$$t_{\bar{\delta}} = \frac{\bar{\delta}}{s_\delta / \sqrt{k}}, \text{ where } \bar{\delta} = \frac{1}{k}\sum_{j=1}^{k}\delta_j, \text{ and } s_\delta = \sqrt{\frac{1}{k-1}\sum_{j=1}^{k}(\delta_j - \bar{\delta})^2}$$

As with the k-fold cross-validation paired t-test, this t-value may then be used as the test statistic in a hypothesis test to distinguish between the error rates of the two algorithms in question.

Essentially, Bouckart (2004, 2005) suggests that based on replicability, type I error, power, and other theoretical considerations, the sorted m×k-fold cross-validation paired t-test is the method of choice. Bouckart (2004) further notes that this test neither requires variance correction (Nadeau and Bengio, 2003), nor does it require calibration for degrees of freedom (Bouckaert, 2003a).

Despite these findings, it is still common to encounter machine learning research that disregards or misuses such statistical tests, though however, there is a growing awareness of their importance and applicability (Demšar, 2006).

[25] Type I error refers to the error of rejecting the null hypothesis when it is in fact true (i.e., a false positive). In typical hypothesis testing, the significance level α equates to the probability of making a type I error (where $0 \leq \alpha \leq 1$).

6.2 Multiple Algorithm Comparisons

When comparing the error rates of a set of algorithms $\boldsymbol{A}$ (where $|\boldsymbol{A}| > 2$), the problem of determining statistical significance is exasperated. More precisely, what is sought is to discover if any algorithm $\boldsymbol{a}_i \in \boldsymbol{A}$, has significantly better performance to another algorithm $\boldsymbol{a}_j \in \boldsymbol{A} \backslash \{\boldsymbol{a}_i\}$. However, at present, there appears to be no hypothesis test that is commonly accepted in the machine learning community for this task (Bouckart, R. R. Personal communication (via email), 2006). The most common approach is to perform pairwise comparisons between each unique pair of algorithms in $\boldsymbol{A}$, and to claim the one with the most (significant) wins to be best (Bouckart, R. R. Personal communication (via email), 2006).

Thus, for example, given some classification dataset $\boldsymbol{s}$, the selection of an algorithm $\boldsymbol{a}_i \in \boldsymbol{A}$ may be performed by first comparing the error rates of each unique pair of algorithms ($\boldsymbol{a}_i$, $\boldsymbol{a}_j$), where $\boldsymbol{a}_i$, $\boldsymbol{a}_j \in \boldsymbol{A}$ and $\boldsymbol{a}_i \neq \boldsymbol{a}_j$ (e.g., via one of the pairwise comparison methods described in Section 6.1), summing the number of wins made by each $\boldsymbol{a}_i \in \boldsymbol{A}$ (across each algorithm pair involving $\boldsymbol{a}_i$), and then selecting the algorithm with the most wins.

7. Summary

This paper serves as an overview of the supervised learning endeavour, and in particular discusses the general task of algorithm selection, and the various methods of evaluating and comparing the generalisation performance of algorithms.

In a classification setting, the problem of algorithm selection relates to the problem of deciding which algorithm $\boldsymbol{a} \in \boldsymbol{A}$ to utilise for a given classification task $\boldsymbol{t}$ (where $\boldsymbol{A}$ corresponds to a subset of applicable candidate algorithms). Given that $\boldsymbol{t}$ is represented by a dataset $\boldsymbol{s}$, the algorithm selection problem thus equates to the search for an algorithm $\boldsymbol{a}$ with the best approximation $\boldsymbol{h}_i \approx \boldsymbol{f}$, where $\boldsymbol{h}_i \in \boldsymbol{H}_{\boldsymbol{a}}(\boldsymbol{s})$ and $\boldsymbol{a}(\boldsymbol{s}) = \boldsymbol{h}_i$, which in turn equates to the search of a candidate algorithm $\boldsymbol{a} \in \boldsymbol{A}$ with the most appropriate inductive bias $\boldsymbol{B}$.

Determining which $\boldsymbol{a} \in \boldsymbol{A}$ achieves the best approximation $\boldsymbol{h}_i \approx \boldsymbol{f}$, and consequently, has the most appropriate inductive bias $\boldsymbol{B}$ relative to the given learning task $\boldsymbol{t}$, depends on the adopted performance metric $\boldsymbol{P}$, where $\boldsymbol{P}$ is some estimator of generalisation performance.

There are many ways to evaluate the generalisation performance of algorithms, most popular among which is 10-fold cross-validation. However, for more significant results, the proper statistical tests should be conducted (e.g., a sorted $m \times k$-fold cross-validation paired t-test for pairwise comparisons). Current research in this area suggests that in order to estimate the performance of an algorithm, the averaged error rate over 10×10-fold stratified cross-validation should be measured, and accordingly, that the sorted 10×10-fold stratified cross-validation paired t-test should be utilized in order to compare the (generalisation) superior of two algorithms.

Therefore, given some classification dataset $\boldsymbol{s}$, the selection of an algorithm $\boldsymbol{a} \in \boldsymbol{A}$ may be performed by first evaluating each $\boldsymbol{a} \in \boldsymbol{A}$ using $\boldsymbol{P}$ (where for example, $\boldsymbol{P}$ corresponds to the averaged error rate over 10×10-fold stratified cross-validation), and then selecting the algorithm with the lowest resultant average error rate. A more statistically sound option, would be to report pairwise comparisons (using the sorted 10×10-fold stratified cross-validation paired t-test), and perhaps to use some aggregation scheme (e.g., summing the number of wins made by some algorithm $\boldsymbol{a}$) to make the selection.

Appendix A

Measuring Confidence Intervals for Error Rate

In order to evaluate the error rate of a learned hypothesis $\boldsymbol{h}$, one typically tests $\boldsymbol{h}$ on a (test) sample $\boldsymbol{S}$ of n instances drawn at random according to the distribution $\boldsymbol{D}$ – this is typically done because the population in question is infinite, and the costs involved in gathering, processing and labelling the data mandates sampling. However, the objective is ultimately to discover the error rate of $\boldsymbol{h}$ over any given instance from the distribution $\boldsymbol{D}$ (not just over $\boldsymbol{S}$).

Let $error_{\boldsymbol{S}}(\boldsymbol{h})$ denote the proportion of incorrect classifications that $\boldsymbol{h}$ makes over $\boldsymbol{S}$, and correspondingly, let $error_{\boldsymbol{D}}(\boldsymbol{h})$ denote the probability that $\boldsymbol{h}$ incorrectly classifies a randomly drawn instance from the distribution $\boldsymbol{D}$. How well does $error_{\boldsymbol{S}}(\boldsymbol{h})$ estimate $error_{\boldsymbol{D}}(\boldsymbol{h})$?

Measuring $error_{\boldsymbol{S}}(\boldsymbol{h})$ is in fact an experiment with a random outcome – i.e., given that $error_{\boldsymbol{S}}(\boldsymbol{h}) = k/n$, since the number of incorrect classifications made by $\boldsymbol{h}$ over $\boldsymbol{S}$, k, is a random variable, it follows that $error_{\boldsymbol{S}}(\boldsymbol{h})$ is also a random variable (Mitchell, 1997). More precisely, a classification of an instance $\boldsymbol{x_i}$ (where $\boldsymbol{x_i} \in \boldsymbol{S}$) by $\boldsymbol{h}$ can be considered a Bernoulli trial[26], in which case – and under certain conditions, measuring $error_{\boldsymbol{S}}(\boldsymbol{h})$ constitutes a Bernoulli process (Witten and Frank, 2000).

A Bernoulli process is a sequence of independent identically distributed trials (Neter et. al., 1988). More specifically, a Bernoulli process is a sequence of trials where: (i) the outcome of each trial is binary, (ii) the outcome of each trial is stationary – i.e., the probability that the outcome is "1" (or inversely, "0"), remains the same from trial to trial, and (iii) the trials are statistically independent[27].

Obviously, each instance $\boldsymbol{x_i}$ may only be classified by $\boldsymbol{h}$ correctly or incorrectly – i.e., the success of a classification by $\boldsymbol{h}$ is binary. However, in most cases, the sampling of $\boldsymbol{S}$ is done without replacement (i.e., upon classification, each instance is not returned to the population for possible re-selection). When sampling is done without replacement, the trials are not independent of each other and the probability of either outcome will vary from trial to trial[28]. Nonetheless, if the sample size is small relative to the size of the population, then for all practical purposes, the trials still constitute a Bernoulli process[29] (Weiss, 1993). A rule of thumb is that a Bernoulli process may be assumed whenever the sample size does not exceed 5% of the population size (Weiss, 1993).

Thus, if $\boldsymbol{S}$ corresponds to a random sample of n independently drawn instances from the distribution $\boldsymbol{D}$, and n is less than 5% of the population size in question (which is typically the case), then the classifications made by $\boldsymbol{h}$ on the n instances in $\boldsymbol{S}$ constitutes a Bernoulli process.

[26] A random trial with two possible outcomes (Weisstein, 2005a; Neter et. al., 1988).
[27] Trials are considered statistically independent if each trial has no bearing on any other trial; i.e., Pr[Trial A | Trial B] = Pr[Trial A] (Weisstein, 2000, 2005d).
[28] If sampling is done with replacement, the trials are independent and identically distributed, and constitute a Bernoulli process.
[29] When trials are dependent, a hypergeometric distribution applies. However, when the size of the population is large relative to the sample, the hypergeometric distribution can be approximated by the binomial distribution (Neter et. al., 1988).

Given a sequence of n classifications (i.e. a Bernoulli process consisting n Bernoulli trials), the number of possible classification sequences[30] (over $\boldsymbol{S}$) containing exactly k incorrect classifications (and thus $n - k$ correct classifications) is given by the binomial coefficient (Choo and Taylor, 1994; Weisstein, 2003a):

$$(A.1) \quad {}^{n}C_{k} = \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Assuming that the exact probability of an incorrect classification (i.e., $error_{D}(\boldsymbol{h})$) is p is known (and thus, that the probability of a correct classification is $1 - p$), the probability of observing $\boldsymbol{h}$ making k incorrect classifications over a sample of n instances (i.e. $Pr[error_{S}(\boldsymbol{h}) = k/n]$, or simply $Pr[k]$) is approximately governed by the binomial distribution[31] – denoted $B(n, p)$, which is defined by the probability function:

$$(A.2) \quad Pr[k] = \binom{n}{k} p^{k} (1-p)^{n-k}$$

Given that the random variable k obeys a binomial distribution, the expected value of k, $E[k] = np$. Consequently, it follows that if n is constant, then $E[k/n] = E[error_{S}(\boldsymbol{h})] = p = error_{D}(\boldsymbol{h})$, and thus, as an estimator[32] of $error_{D}(\boldsymbol{h})$, $error_{S}(\boldsymbol{h})$ is unbiased[33] – it is also efficient[34], or rather, will yield the smallest expected squared error between the estimate and the true value of the parameter (Mitchell, 1997).

Correspondingly, the standard deviation of the variable k (i.e., the standard deviation of a random variable governed by the binomial distribution $B(n, p)$) is:

$$(A.3) \quad \sigma[k] = \sqrt{np(1-p)}$$

Further, given that n is constant, the standard deviation of $error_{S}(\boldsymbol{h})$ is thus:

$$(A.4) \quad \sigma[error_{S}(\boldsymbol{h})] = \sigma[k/n] = \frac{\sigma[k]}{n} = \frac{\sqrt{np(1-p)}}{n} = \sqrt{\frac{p(1-p)}{n}}$$

Unfortunately, p is unknown. Fortunately however, $error_{S}(\boldsymbol{h}) = k/n$ may be substituted for p (Mitchell, 1997; Weiss, 1993) – since k/n is an unbiased and efficient estimator for p.

Thus, since the probability distribution function of $error_{S}(\boldsymbol{h})$ is known, with some specified confidence α, the range in which $error_{D}(\boldsymbol{h})$ should lie may be computed. However, when n is large, the resultant factorials in the probability function of the binomial distribution become cumbersome to compute.

Fortunately, there is a normal approximation to the binomial distribution. In fact, the normal distribution[35] was first introduced by Abraham de Moivre in an article in 1733 (and later reprinted in the later editions of his The Doctrine of Chances – in 1738 and 1756) in the context of approximating certain binomial distributions for large n (Daw and Pearson, 1972). More

[30] Let a classification sequence correspond to the sequence of graded classifications made by some $\boldsymbol{h}$; e.g., "11110" corresponds to a classification sequence where $\boldsymbol{h}$ correctly classified the first four instances, but incorrectly classified the fifth and last instance.

[31] A plethora of literature exists on the binomial distribution – e.g., refer to (Weisstein, 2002a).

[32] In general, an estimator is any random variable used to estimate some parameter (Weisstein, 2003b) of the underlying population from which the sample is drawn (Mitchell, 1997). Also refer to (Weisstein, 2005c).

[33] An unbiased estimator (Weisstein, 1999) is an estimator with zero estimation bias (Weisstein, 2002c).

[34] Not to be confused with computational efficiency, an unbiased estimator $\boldsymbol{Q_1}$ has greater efficiency than another unbiased estimator $\boldsymbol{Q_2}$ if $\sigma^2[\boldsymbol{Q_1}] < \sigma^2[\boldsymbol{Q_2}]$ (Neter et. al., 1988).

[35] Literature on the normal distribution is as, if not more abundant than the binomial distribution's – e.g., refer to Weisstein (2004a).

contemporarily seen as a consequence of the central limit theorem (Weisstein, 2005b), essentially if n is large enough, and the skew of the distribution is not too great, then an approximation (provided a suitable continuity correction is used) to $B(n, p)$ is given by the normal distribution, denoted $N(np, np(1 - p))$ (Devroye, 1986). As a rule of thumb, if both np and $n(1 - p)$ are greater than 5, then the normal approximation is suitable (Weiss, 1993; Devroye, 1986).

Therefore, suppose that over a (test) sample $\boldsymbol{S}$ consisting n instances (randomly drawn from the distribution $\boldsymbol{D}$ – with or without replacement), the hypothesis $\boldsymbol{h}$ makes k incorrect classifications. Then since $errors(\boldsymbol{h})$ approximately follows the normal distribution, a confidence interval[36] for $error_{\boldsymbol{D}}(\boldsymbol{h})$ may calculated by utilising our estimate for p, k/n, and the areas under the standard normal curve – Figure A.1 depicts the standard normal curve.

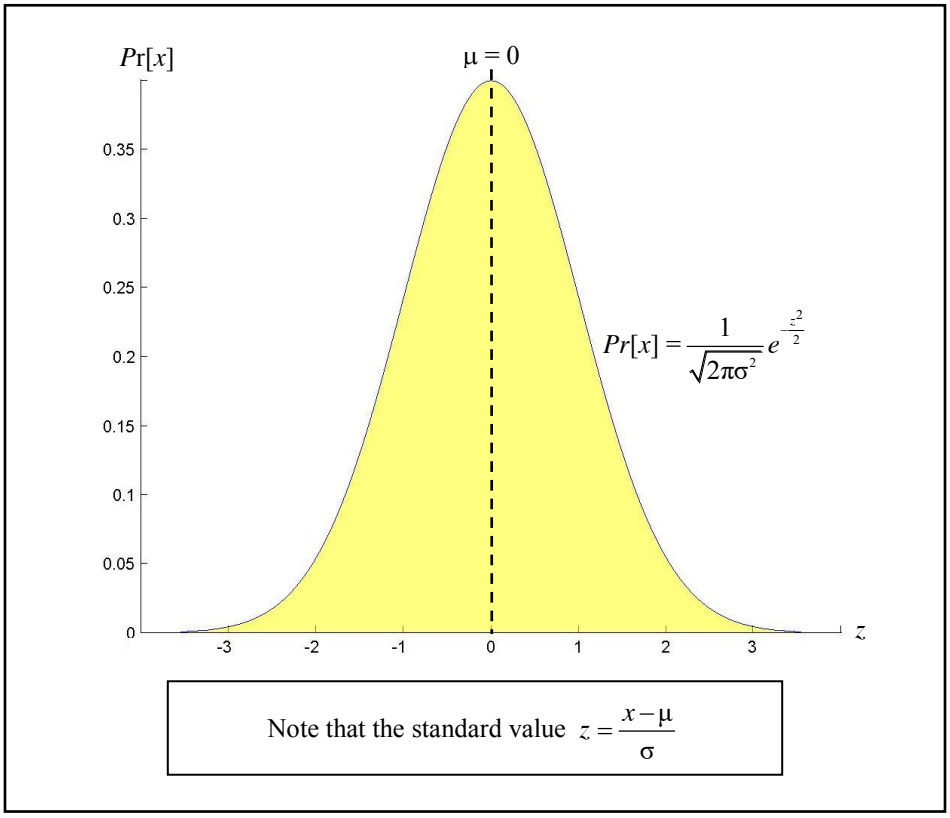

Figure A.1 – The standard normal curve $N(0,1)$.

The area under a probability distribution pertains to all the possible values and corresponding frequencies of the random variable in question, and as such, the total area under any probability distribution is equal to 1. In the case of the standard normal distribution $N(0, 1)$, $100\alpha\%$ (where $0 \leq \alpha \leq 1$) of the area under the curve centred around 0 (i.e., its maxima) can be specifically computed, and is tabulated in most statistical texts – e.g. see Appendix C in Neter et. al. (1988). Essentially, such tables list the interval from 0 to a standardised value z (where $z = (x - \mu) / \sigma$) in one direction that corresponds to a certain percentage of the area underneath $N(0, 1)$ – e.g.,

[36] A confidence interval denotes the range in which a parameter p sits with probability α ($1 - \alpha$ is typically known as the confidence coefficient) (Weisstein, 2002b; Neter et. al., 1988).

approximately 47.5% of the area under $N(0, 1)$ lies in the z-value interval 0 to 1.96. Accordingly, since the standard normal curve is symmetric, these intervals may be extended in both directions away from 0 – e.g., approximately 95% of the area under $N(0, 1)$ lies in the z-value interval –1.96 to 1.96. Table A.1 summaries the various z-value intervals and their corresponding areas under $N(0, 1)$.

Standard value $z_{\alpha/2}$:	0.67	1.15	1.64	1.96	2.58	2.81	3.30
$((\alpha/2) \times 100)\%$ of the area from $-z_{\alpha/2}$ to 0 or from 0 to $z_{\alpha/2}$:	25.0	37.5	45.0	47.5	49.5	49.75	49.95
$(\alpha \times 100)\%$ of the area from $-z_{\alpha/2}$ to $z_{\alpha/2}$:	50.0	75.0	90.0	95.0	99.0	99.5	99.9

Table A.1: The standard values $z_{\alpha/2}$ corresponding to $((\alpha/2) \times 100)\%$ and $(\alpha \times 100)\%$ of the areas underneath standard normal curve $N(0,1)$ ranging from $-z_{\alpha/2}$ to $z_{\alpha/2}$ and 0 to $z_{\alpha/2}$ (or equivalently, to $-z_{\alpha/2}$ to 0) respectively.

Recall that $error_S(\boldsymbol{h})$ is an unbiased estimator of $error_D(\boldsymbol{h})$, and that $E[error_S(\boldsymbol{h})] = error_D(\boldsymbol{h})$. Further, given that $error_S(\boldsymbol{h})$ approximately follows the normal distribution, one can expect that $error_D(\boldsymbol{h})$ will fall within $error_S(\boldsymbol{h}) \pm (z_{\alpha/2} \times \sigma[error_S(\boldsymbol{h})])$ with $100\alpha\%$ probability. Essentially, the various possible values for $error_S(\boldsymbol{h})$ (given various $\boldsymbol{S}$ randomly drawn via $\boldsymbol{D}$ independent of $\boldsymbol{h}$) will follow the normal distribution, then with $100\alpha\%$ probability, the standardised values for $error_S(\boldsymbol{h})$ will fall in the interval $-z_{\alpha/2}$ to $z_{\alpha/2}$ (as given in Table A.1).

In summary, if $\boldsymbol{S}$ corresponds to a random sample of n independently drawn instances from the distribution $\boldsymbol{D}$, $n = |\boldsymbol{S}|$, n is less than 5% of the population size in question, n is large enough, and the skew of the distribution is not too great (e.g., if both np and $n(1 - p)$ are greater than 5), then with $100\alpha\%$ probability, $error_D(\boldsymbol{h})$ will fall within the interval given by:

$$error_S(\boldsymbol{h}) \pm z_{\alpha/2}\sqrt{\frac{error_S(\boldsymbol{h})(1-error_S(\boldsymbol{h}))}{n}}$$

Appendix B

The McNemar Test and 5×2-fold Cross-validation Paired *t*-test

In McNemar's test, the available data ***s*** is divided into a training set ***s-train*** and test set ***s-test*** (e.g., as per the typical holdout procedure). The two algorithms under scrutiny, $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are trained with ***s-train***, and the resultant hypotheses (i.e., $\boldsymbol{a}_1$(***s-train***) and $\boldsymbol{a}_2$(***s-train***)) consequently evaluated via ***s-test***. The contingency table described in table B.1 is then constructed.

N_{00} = Number of examples in ***s-test*** misclassified by $\boldsymbol{a}_1$(***s-train***) and $\boldsymbol{a}_2$(***s-train***)	N_{01} = Number of examples in ***s-test*** misclassified by $\boldsymbol{a}_1$(***s-train***) but not $\boldsymbol{a}_2$(***s-train***)
N_{10} = Number of examples in ***s-test*** misclassified by $\boldsymbol{a}_2$(***s-train***) but not $\boldsymbol{a}_1$(***s-train***)	N_{11} = Number of examples in ***s-test*** correctly classified by $\boldsymbol{a}_1$(***s-train***) and $\boldsymbol{a}_2$(***s-train***)

Table B.1: The contingency table used in McNemar's test. This table represents all the possible classification patterns over the hypotheses $\boldsymbol{a}_1$(***s-train***) and $\boldsymbol{a}_2$(***s-train***) when evaluated on ***s-test***.

Let H_0 correspond to the hypothesis that both algorithms have the same error rate, and that H_a correspond to the hypothesis that both algorithms have different error rates. If H_0 is true, then $N_{01} = N_{10}$. McNemar's test utilises a χ^2 goodness-to-fit test that compares the distribution of counts expected under H_0 – given in table B.2, to the counts observed during testing.

N_{00}	$(N_{01} + N_{10})/2$
$(N_{01} + N_{10})/2$	N_{11}

Table B.2: The expected contingency table under H_0, which hypotheses that $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ should have the same error rate. Accordingly, since this is a hypothesis that $N_{01} = N_{10}$, one should thus expect that the number of instances that only particular algorithm (i.e., $\boldsymbol{a}_i$(***s-train***)) misclassifies will average out (over many test sets) to be equal to the number of instances that the other algorithm (i.e., $\boldsymbol{a}_j$(***s-train***)) misclassifies.

The following statistic may thus be computed:

$$\frac{(|N_{01} - N_{10}| - 1)^2}{N_{01} + N_{10}}$$

This statistic[37] approximately follows the χ^2 distribution with 1 degree of freedom. Correspondingly, if H_0 is true, with $100\alpha\%$ probability, the measured statistic should be less

[37] Note that it incorporates a continuity correction in the form of the –1 in the numerator to account for the fact that the statistic is discrete while the χ^2 distribution is continuous.

than $\chi^2(1, 1 - \alpha)$, which is the value given by the χ^2 distribution with 1 degree of freedom and significance $(1 - \alpha)$.

With the 5×2-fold cross-validation paired *t*-test, 5 runs of 2-fold cross-validation are performed, where each fold corresponds to the random division of the given dataset ***s***, into 2 subsets ***s-f1*** and ***s-f2***. For each of the 5 (***s-f1***, ***s-f2***) pairs, 4 error rates are computed using the two algorithms $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$: $\boldsymbol{error}_{s\text{-}f1}(\boldsymbol{a}_1(\boldsymbol{s\text{-}f2}))$, $\boldsymbol{error}_{s\text{-}f1}(\boldsymbol{a}_2(\boldsymbol{s\text{-}f2}))$, $\boldsymbol{error}_{s\text{-}f2}(\boldsymbol{a}_1(\boldsymbol{s\text{-}f1}))$, and $\boldsymbol{error}_{s\text{-}f2}(\boldsymbol{a}_2(\boldsymbol{s\text{-}f1}))$. Consequently, one can compute the variance of the differences in error rate for the *i*-th run, $s^2[d_i] = (d_{i.1} - \bar{d}_i)^2 + (d_{i.2} - \bar{d}_i)^2$, where the mean difference of the *i*-th run, $\bar{d}_i = (d_{i.1} + d_{i.2})/2$, and where the two differences for the *i*-th run are $d_{i.1} = \boldsymbol{error}_{s\text{-}f1}(\boldsymbol{a}_1(\boldsymbol{s\text{-}f2})) - \boldsymbol{error}_{s\text{-}f1}(\boldsymbol{a}_2(\boldsymbol{s\text{-}f2}))$, and $d_{i.2} = \boldsymbol{error}_{s\text{-}f2}(\boldsymbol{a}_1(\boldsymbol{s\text{-}f1})) - \boldsymbol{error}_{s\text{-}f2}(\boldsymbol{a}_2(\boldsymbol{s\text{-}f1}))$.

Dietterich (1998) defines the 5×2cv statistic as:

$$\tilde{t} = \frac{d_{1.1}}{\sqrt{\frac{1}{5}\sum_{i=1}^{5} s^2[d_i]}}$$

and shows that under H_0, $\tilde{t}$ has approximately the *t* distribution with 5 degrees of freedom – a *t*-test on $\tilde{t}$ can thus be performed to determine if $\boldsymbol{a}_1$ is better than $\boldsymbol{a}_2$.

References

1. Bailey, T., and Elkan, C. (1993). Estimating the Accuracy of Learned Concepts. In Bajcsy, R. (Ed.), Proceedings of the 13th International Joint Conference on Artificial Intelligence, 895-900.

2. Bensusan, H. (1999). Automatic Bias Learning: An Inquiry into the Inductive Bias of Induction. PhD Thesis, School of Cognitive and Computing Sciences, University of Sussex.

3. Breiman, L., and Spector, P. (1992). Submodel Selection and Evaluation in Regression: The X-random Case. International Statistical Review, 60(3):291-319.

4. Bouckaert, R. (2003a). Choosing between Two Learning Algorithms based on Calibrated Tests. In Proceedings of the 20th International Conference of Machine Learning, 51-58.

5. Bouckaert, R. (2003b). Choosing Learning Algorithms Using Sign Tests with High Replicability. In Proceedings of the 16th Australian Conference on Artificial Intelligence, 710-722.

6. Bouckaert, R. (2004). Estimating Replicability of Classifier Learning Experiments. In Proceedings of the 21st International Conference on Machine Learning.

7. Bouckaert, R. (2005). Low Replicability of Machine Learning Experiments is not a Small Data Set Phenomenon. In Proceedings of the 22nd International Conference on Machine Learning, Workshop on Meta-learning, 4-11.

8. Bouckaert, R., and Frank, E. (2004). Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms. In Proceedings of the Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, 3-12.

9. Choo, K., and Taylor, D. (1994). Introduction to Discrete Mathematics. Addison Wesley Longman.

10. Daw, R., and Pearson, E. (1972). Studies in the History of Probability and Statistics XXX: Abraham de Moivre's 1733 Derivation of the Normal Curve: a Bibliographical Note. Biometrika, 59:677-680.

11. Demšar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research, 7:1-30.

12. DesJardins, M., and Gordon, D. (1995). Evaluation and Selection of Bias in Machine Learning. Machine Learning, 20(1-2):5-22.

13. Devroye, L. (1986). Non-Uniform Random Variate Generation. Springer-Verlag.

14. Dietterich, T. (1990). Machine Learning. Annual Review of Computer Science, 4:255-306.

15. Dietterich, T., and Kong, E. (1995). Machine Learning Bias, Statistical Bias, and Statistical Variance of Decision Tree Algorithms. Technical Report, Department of Computer Science, Oregon State University.

16. Dietterich, T. (1997). Fundamental Experimental Research in Machine Learning. In J. McCarth (Ed.), Basic Topics in Experimental Computer Science.
[http://web.engr.oregonstate.edu/~tgd/experimental-research/index.html]
Accessed 2006 Jan 17.

17. Dietterich, T. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Computation, 10:1895-1924.

18. Dietterich, T. (2003). Machine Learning. In Nature Encyclopedia of Cognitive Science. Macmillan.

19. Efron, B. (1979). Computers and the Theory of Statistics: Thinking the Unthinkable. Society for Industrial and Applied Mathematics Review, 21:460-480.

20. Efron, B. (1983). Estimating the Error Rate of a Prediction Rule: Improvements on Cross-validation. Journal of the American Statistical Association, 78(382):316-330.

21. Efron, B., and Tibshirani, R. (1993). An Introduction to the Bootstrap. Chapman and Hall.

22. Everitt, B. (1977). The Analysis of Contingency Tables. Chapman and Hall.

23. Freund, Y., and Schapire, R. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences, 55(1):119-139.

24. Giraud-Carrier, C. (1998). Beyond Predictive Accuracy: What? In Proceedings of the 10th European Conference on Machine Learning, Workshop on Upgrading Learning to the Meta-Level: Model Selection and Data Transformation, 78-88.

25. Giraud-Carrier, C. and Provost, F. (2005). Toward a Justification of Meta-learning: Is the No Free Lunch Theorem a Show-stopper? In Proceedings of the 22nd International Conference on Machine Learning, Workshop on Meta-learning, 12-19.

26. Guyon, I., and Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3:1157-1182.

27. Hastie, T., Tibshirani, R., and Friedman, J. (2001). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag.

28. Henery, R. (1994). Methods for Comparison. In Michie, D., Spiegelhalter, D., and Taylor, C. (Eds.), Machine Learning, Neural and Statistical Classification. Ellis Horwood.

29. Hume, D. (1740). A Treatise of Human Nature. Beauchamp, T. (Ed.), Clarendon Press, 2001.

30. Hume, D. (1748). An Enquiry Concerning Human Understanding. Edited by Beauchamp, T. L., Clarendon Press, 2006.

31. Keller, J., Paterson, I., and Berrer, H. (2000). An Integrated Concept for Multi-criteria-ranking of Data-mining Algorithms. In Proceedings of the 11th European Conference on Machine Learning, Workshop on Meta-learning: Building Automatic Advice Strategies for Model Selection and Method Combination, 73-86.

32. Kneale, W. (1949). Probability and Induction. Clarendon Press.

33. Kohavi, R. (1995). A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In Mellish, C. (Ed.), Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1137-1143.

34. Lachenbruch, P., and Mickey, M. (1968). Estimation of Error Rates in Discriminant Analysis. Technometrics, 10:1-11.

35. Langford, J. (2005a). The Cross Validation Problem. In Proceedings of the 18th Annual Conference on Learning Theory, 687-688.

36. Langford, J. (2005b). Tutorial on Practical Prediction Theory for Classification. Journal of Machine Learning Research, 6: 273-306.

37. Mitchell, T. (1990). The Need for Biases in Learning Generalizations. In Shavlik, J., and Dietterich, T. (Eds.), Readings in Machine Learning, 185-191. Morgan Kaufmann.

38. Mitchell, T. (1997). Machine Learning. McGraw-Hill.

39. Nadeau, C., and Bengio, Y. (2003). Inference for the Generalization Error. Machine Learning, 52(3):239-281.

40. Nakhaeizadeh, G., and Schnabel, A. (1997). Development of Multi-criteria Metrics for Evaluation of Data Mining Algorithms. In Proceedings of the 3rd Conference on Knowledge Discovery and Data-mining, 37-42.

41. Neter, J., Wasserman, W., and Whitmore, G. (1988). Applied Statistics (3rd Ed.). Allyn and Bacon.

42. Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann.

43. Rao, R., Gordon, D., and Spears, W. (1995). For Every Generalization Action, Is There Really an Equal and Opposite Reaction? Analysis of the Conservation Law for Generalization Performance. In Proceedings of the 12th International Conference on Machine Learning, 471-479.

44. Rendell, L. (1986). A General Framework for Induction and a Study of Selective Induction. Machine Learning, 1(2): 177-226.

45. Schaffer, C. (1994). A Conservation Law for Generalization Performance. In Cohen, W., and Hirsh, H. (Eds.), Proceedings of the 11th International Conference on Machine Learning, 259-265.

46. Soares, C., Costa, J., and Brazdil, P., (2000). A Simple and Intuitive Measure for Multicriteria Evaluation of Classification Algorithms. In Proceedings of the 11th European Conference on Machine Learning, Workshop on Meta-learning: Building Automatic Advice Strategies for Model Selection and Method Combination, 87-96.

47. Stone, M. (1974). Cross-validatory Choice and Assessment of Statistical Predictions (with discussion). Journal of the Royal Statistical Society, Series B, 36:111-147.

48. Stone, M. (1977). Asymptotics for and against Cross-validation. Biometrika, 64:29-35.

49. Sutton, R., and Barto, A. (1998). Reinforcement Learning: An Introduction. The MIT Press.

50. Utgoff, P. (1986). Shift of Bias for Inductive Concept Learning. In Michalski, R., Carbonell, J., and Mitchell, T. (Eds.), Machine Learning: An Artificial Intelligence Approach, Vol. II:107-148. Morgan Kaufmann.

51. Weiss, N. A. (1993). Elementary Statistics (2nd Ed.). Addison-Wesley.

52. Weiss, S. M., and Indurkhya, N. (1994). Decision Tree Pruning: Biased or Optimal. In Proceedings of the 12th National Conference on Artificial Intelligence, 626-632.

53. Weisstein, E. (1999). Unbiased Estimator. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/ UnbiasedEstimator.html] Accessed 2006 Jan 20.

54. Weisstein, E. (2000). Independent Events. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/BernoulliTrial.html] Accessed 2006 Jan 20.

55. Weisstein, E. (2001). Student's t-Distribution. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/Studentst-Distribution.html] Accessed 2006 Jan 20.

56. Weisstein, E. (2002a). Binomial Distribution. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/BinomialDistribution.html] Accessed 2006 Jan 20.

57. Weisstein, E. (2002b). Confidence Interval. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/ConfidenceInterval.html] Accessed 2006 Jan 20.

58. Weisstein, E. (2002c). Estimator Bias. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/EstimatorBias.html] Accessed 2006 Jan 20.

59. Weisstein, E. (2003a). Binomial Coefficient. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/BinomialCoefficient.html] Accessed 2006 Jan 20.

60. Weisstein, E. (2003b). Parameter. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/Parameter.html] Accessed 2006 Jan 20.

61. Weisstein, E. (2004a). Normal Distribution. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/NormalDistribution.html] Accessed 2006 Jan 20.

62. Weisstein, E. (2005a). Bernoulli Trial. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/BernoulliTrial.html] Accessed 2006 Jan 20.

63. Weisstein, E. (2005b). Central Limit Theorem. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/CentralLimitTheorem.html] Accessed 2006 Jan 20.

64. Weisstein, E. (2005c). Estimator. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/Estimator.html] Accessed 2006 Jan 20.

65. Weisstein, E. (2005d). Independent Statistics. In MathWorld - A Wolfram Web Resource. [http://mathworld.wolfram.com/IndependentStatistics.html] Accessed 2006 Jan 20.

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

67. Wolpert, D. (1996a). The Existence of Apriori Distinctions between Learning Algorithms. Neural Computation, 8:1391-1420.

68. Wolpert, D. (1996b). The Lack of Apriori Distinctions between Learning Algorithms. Neural Computation, 8:1341-1390.

69. Wolpert, D. (2000). Any Two Learning Algorithms Are (Almost) Exactly Identical. In Proceedings of the 17th International Conference on Machine Learning, Workshop on What Works Well Where.

70. Wolpert, D. (2001). The Supervised Learning No-free-lunch Theorems. In Roy, R., Koppen, M., Ovaska, S., Fu-ruhashi, T., and Hoffman, F. (Eds.), Soft Computing in Industry: Recent Applications, 25-42. Springer-Verlag.