ARTIFICIAL INTELLIGENCE, GENERALIZATION

Generalization in psychology is the tendency to respond in the same way to different but similar stimuli (1). Such transfer of tendency may be based on temporal stimuli, spatial cues, or other physical characteristics. Learning, on the other hand, may be considered as a balance between generalization and discrimination (the ability to respond to differences among stimuli). An imbalance between them may lead to negative results. A system that discriminates but does not generalize does not learn. Because it is unable to learn, it may not be able to respond similarly to stimuli with small differences. Likewise, a system that generalizes but does not discriminate may respond similarly all the time.

Machine learning is an area in artificial intelligence that extends knowledge, concepts, and understanding through one or more observations of instances of the concept (2). The number of instances involved and the amount of information they carry will determine the learning method to be used.

Learning methods can be classified as data-intensive and knowledge-intensive (see Fig. 1). In *data-intensive methods,* symbolic concepts are learned using data-intensive similarity-based methods. The learner is shown a large number of related examples and is required to identify their similarities and generalize the concept embedded. Using this approach, Mitchell (3) defines *generalization* as a process that takes into account a large number of specific observations (inductive bias), and that extracts and retains the important features that characterize classes of these observations. He then casts generalization as a search problem, and alternative generalization methods as different search strategies.

An example of a data-intensive learning method is the learning of heuristics represented as production rules (4). In this approach, a heuristic method is represented as a collection of production rules, and learning modifies these rules based on positive and negative examples and on decisions made in these rules. The process of apportioning a feedback signal to individual decisions carried out in the past, as well as to decision elements applied in each decision, in order to refine the heuristic method is called *credit assignment.* The former credit assignment is called *temporal,* and the latter, *structural.* Credit assignment is usually difficult when learning incrementally single concepts from examples, especially when learning multiple disjunctive concepts and when the learning data is noisy. In this case, a teacher may be needed to tell the learner the proper amount of credit to assign to a decision.

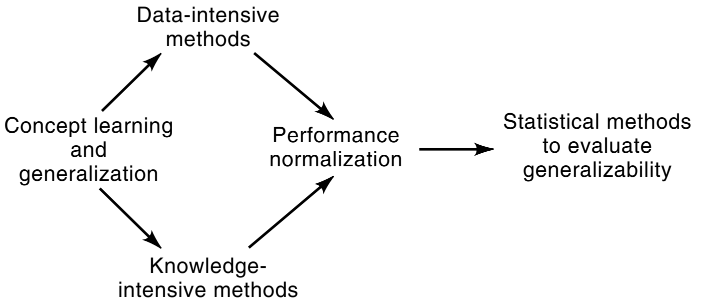

Figure 1. The relationship between concept learning, generalization, and generalizability.

A second class of data-intensive learning methods are *decision-theoretic methods* that use statistical decision theory to discriminate probabilistic patterns exhibited in learning examples (5). The major component in a decision-theoretic approach is the *loss function* that measures the loss when the learner categorizes a learning example incorrectly. It represents a statistical approach to credit assignment. By minimizing the total loss using statistical methods, it is sometimes possible to show asymptotic convergence of the concept to be learned. Examples of decision-theoretic methods include evolutionary programming (6), genetic algorithms (7), classifier systems (8), and artificial neural networks (ANNs) (9).

In contrast to using extensive training examples in data-intensive methods, *knowledge-intensive methods* rely on domain-specific knowledge to learn and to generalize. In *explanation-based learning,* the learner analyzes a single training example using domain knowledge and the concept under study to produce a generalization of the example and a deductive justification of the generalization (10,11). Knowledge-intensive methods work well when the concept to be generalized can be deduced from the domain knowledge.

To evaluate the quality of a learning and generalization method and to measure the degree to which learning and generalization has been achieved, generalizability measures have been developed. In the simplest case, they measure the number of positive and negative examples in learning. In more general cases, the degree to which an example satisfies a learned concept must be considered, and statistical techniques are employed to determine whether a learned concept can be generalized.

For example, in learning in feedforward ANNs, the effectiveness of an ANN that computes discrete {0,1}-valued mappings can be evaluated by the network's ability to solve dichotomization problems using measures such as discrimination capacity, VC-dimension (named after Vapnik and Chervonenkis), and efficiency of decision functions (12). For an ANN that performs function approximation computing either discrete multiple-valued or continuous mappings, we can measure its quality using concepts such as combinatorial dimension, approximation error, and estimation error. Finally, the concept of PAC (probably approximately correct) learning (13) is useful for characterizing the time complexity of algorithms for learning both discrete and continuous mappings.

A related problem in generalizability is the normalization of learned results relative to a baseline. When the quality of a learned concept is measured numerically and depends on some attributes of the example, it may be necessary to normalize the measure with respect to that of a baseline before

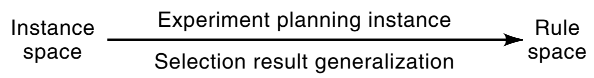

Figure 2. The process of inductive learning and generalization.

any statistical evaluations can be made. For instance, the quality measure of a learned concept may depend on the size of the learning example and needs to be normalized before results from multiple learning examples can be aggregated statistically. In this case, the generalizability of the learned concept may depend on the baseline and the statistical method used to aggregate performance measures. Anomalies in the ordering of hypotheses may happen when different normalization and aggregation methods are used. This is discussed in detail in a later section.

In the next section we summarize previous approaches in generalization and credit assignment. We then present the general concept of generalizability, generalizability measures, and anomalies in generalization when performance measures are normalized and aggregated.

CONCEPT GENERALIZATION USING INDUCTION

In this section we summarize various strategies for generalization. Early work on inductive learning and generalization was done by Simon and Lea (14) who used training instances selected from some space of possible instances to guide the search for general rules. The process of inductive learning entails a mapping from the instance space to the rule space and involves experiment planning, instance selection, and result interpretation (or generalization). (See Fig. 2.) Here, a set of problem instances are used to guide the selection of a set of rules in the rule space that generalize the instances. The resulting set of rules may be organized as distinct rules or as decision trees.

The Generalization Problem

Generalization involves the extraction of information useful to guide the search of a rule space (2). To simplify the search process, a good representation of the rule space must be chosen so that generalization can be carried out by inexpensive syntactic operations, such as turning constants to variables, dropping conditions, adding options, curve fitting, and zeroing a coefficient.

The specific operators used may depend on the representation of the rule space. For instance, a production rule $Z \rightarrow Z'$ can be used to represent either the backward form (Z is the input condition, and Z' is the value of a state vector plus associated predicate) or the forward form (Z' is a computational rule). The evaluation of the execution of a rule constitutes credit assignment, whereas the creation of new rules involves generalization. The latter entails the identification of a subvector of variables relevant to the creation, the proper decision for the situation, and the reason for making the decision. Waterman (4) proposed a set of generalization operators that modify the defined symbolic values in a rule, eliminate one or more variables in a rule, and change action rules and error-causing rules.

Mitchell (3) defines generalization in the context of a language that describes instances and generalizations. Based on a set of positive and negative examples, predicates are matched from generalizations to instances. Hence, generalizations are defined within the provided language that are consistent with the presented training examples.

In general, generalization also requires a function to evaluate the positive and negative examples obtained in order to provide feedback (credit assignment). In the simplest case, the function counts the number of positive and negative examples. In decision-theoretic approaches, a loss function is used to measure the loss when the learner categorizes a learning example incorrectly. This is the approach taken in classifier-system and genetics-based learning that uses a fitness function. In reinforcement learning, the evaluation function may have to be learned independently in order to provide proper temporal credit assignment. This is the approach taken in learning an ANN for pole balancing (15) truck backer (16), and neural-network design (17). The reinforcement function is particularly difficult to design when examples drawn from the problem space are not statistically related. This happens when the evaluation data depends on the size of the examples, or when the examples drawn belong to different problem subdomains. Some possible solutions to these issues are discussed in a later section.

Generalization Strategies

As defined by Mitchell (3,11), generalization strategies can broadly be classified as data driven and knowledge-driven. (See Fig. 3.) Both paradigms use generate-and-test that generates alternative concepts, tests them on test cases, and constructs feedbacks (credit assignment) to aid the refinement of the concepts generated. The difference lies in the amount of tests performed: data-driven methods do not rely on domain knowledge and often require extensive tests on the concepts under consideration before reliable feedbacks can be generated. In contrast, knowledge-driven methods rely on domain knowledge and one or a few tests to deduce new concepts.

Data-driven generalization strategies can be classified into depth-first search, breadth-first search, version-space, and decision-theoretic techniques (3).

A *depth-first strategy* starts from a single generalization as the current best hypothesis, tests it against each training example, and modifies the hypothesis in order to make it consistent with the training example. Its advantage is that it keeps a global picture in mind when modifying the hypothesis. However, it is usually expensive to backtrack when a negative training example is found. In this case, the new hypothesis generated must be tested against all previous training examples to make sure that they are consistent. Any inconsistencies will incur further backtracking.

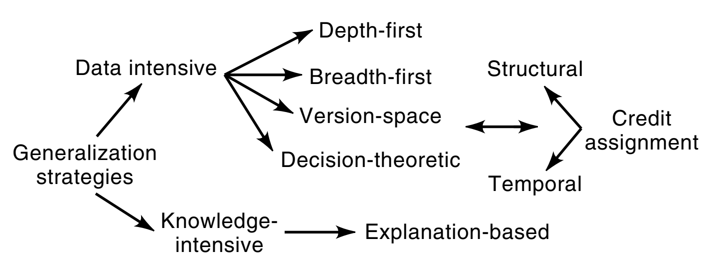

Figure 3. A classification of generalization strategies.

A *breadth-first strategy,* on the other hand, generalizes from more specific hypotheses to more general ones. Initially, it starts from a set of the most specific hypotheses. Positive training examples allow the search to progress down the breadth-first tree, generating more general hypotheses, whereas negative training examples will prune the corresponding hypothesis from the search tree. The boundary of the search tree, therefore, represents the most general hypotheses generated so far that are consistent with the (positive) training examples. As a result, when a new (more general) hypothesis is generated, it only needs to be tested against all positive training examples to make sure that they are consistent with the current hypothesis. This is the main advantage of a breadth-first search over a depth-first search.

A hybrid of depth-first and breadth-first strategies is a *version-space* strategy. The version space represents the set of hypothesis that are consistent with all the training examples. It defines two boundaries. The first boundary is obtained by depth-first search and bounds the acceptable level of specialization of hypotheses (those that are consistent with all the positive examples). The second boundary is obtained by breadth-first search and bounds the acceptable level of generality of hypotheses (those that are inconsistent with all the negative examples).

A third class of data-driven generalization strategies are the decision-theoretic techniques. These do not always use a single type of search method but may use a hybrid of search methods, such as depth-first and breadth-first searches, depending on the evaluation results. They rely of a loss function that measures the expected loss when the learner categorizes a learning example incorrectly. Although the loss function may be designed either based on formal statistical methods or heuristically, the generalization strategy can generally be shown to converge asymptotically to the desired concept. For instance, in genetic algorithms, Holland's Schema Theorem (7) shows that the number of structures in a knowledge base that share a given subset of components can be expected to increase or decrease over time at a rate proportional to the observed performance of the subset, eventually converging asymptotically to the optimal configuration.

In contrast to data-driven techniques, an explanation-based generalization strategy uses domain knowledge to generalize from an example, defining a concept that contains the example (11). It analyzes a single example in terms of the domain knowledge and the goal concept and produces a proof (or explanation) that shows that the example is an instance of the goal concept. Here, the goal concept found satisfies the operationality criteria, which is a predicate over concept definitions that specifies the form in which the concept must be learned. The proof tree in the process of generalization is constructed by replacing each instantiated rule by the associated general rule.

An explanation-based strategy can start from general concepts to derive specific ones, or vice versa. It consists of two phases: explanation and generalization. In the explanation phase, the relevant features of the training example are isolated in order to create an explanation structure that terminates in an expression satisfying the operationality criterion. In the generalization phase, a set of sufficient conditions are found to satisfy the explanation. This is done by regressing the goal concept through the explanation structure and by composing terms from different parts of the explanation to form a valid generalization.

One of the problems in explanation-based learning is that learning through multiple examples may result in multiple rules that cannot be combined into a single rule. This leads to gradual degradation in efficiency in the generalized rules. Another problem is that explanation-based generalization does not create new parameters; hence, parameters not explicitly expressed in the proof cannot be generalized. In this context, studies have been made to generalize the structure of an example proof such that a fixed number of rule applications in the proof is generalized into an unbounded number of applications (18).

Credit Assignment

Credit assignment entails the apportioning of feedback signals to individual decisions made in the past as well as rules/entities leading to a decision. The application of credit assignment requires a world model that captures the relationship among states, decisions, and feedback signals generated by the learning system or measured in the environment. This world model is explicitly defined in knowledge-rich applications, but may have to be inferred during learning and generalization when domain knowledge is not available. Credit assignment is further complicated when there may be delays in getting the feedback signals due to a decision. In this case, multiple subsequent decisions may have been made between the times a decision was made and its feedback signal received.

There are two types of credit assignment: structural and temporal (15). *Structural credit assignment* entails ways of using feedback signals to refine the individual components or rules of a hypothesis. This process is systematic in explanation-based learning as it involves rewriting one rule into another in the proof structure. In other learning approaches, credit assignment may not be possible when domain knowledge is missing. In this case, one may employ population-based learning (19) that maintains a population of competing hypotheses and delays choosing the best hypothesis to evaluate or new ones to create until more tests are performed on each of the alternatives.

Temporal credit assignment, on the other hand, entails the apportioning of temporal global feedback signals by the learning system to the past decisions that affect these signals. When a decision is applied, its temporal scope is the interval of time during which its direct effect can be observed in the application environment. If the temporal scope is infinite and state changes are Markovian, then the effects due to a feedback signal will be attributed only to the most recent decision made in the past, and the effects of other decisions will be felt indirectly through intervening decisions and states. When the temporal scope is finite and state changes are dependent and non-Markovian, then an approximate temporal model is needed for temporal credit assignment. Temporal credit assignment is used extensively in reinforcement learning (15).

Credit assignment can be either implicit or explicit. An example of implicit credit assignment is done in LS-1 (20) in which rules that are physically close together on the list representing the knowledge structure stand a good chance of being inherited as a group. On the other hand, in explicit credit assignment, explicit rules are defined for credit assignment.

Examples of explicit temporal credit-assignment mechanisms are the profit sharing plan and the bucket brigade algorithm in classifier systems (21). A hybrid of implicit and explicit credit assignment can also be defined (22).

GENERALIZABILITY MEASURES

To evaluate whether the goal of learning and generalization is achieved, generalizability measures are used to evaluate the quality of generalization. These measures are not limited to the field of machine learning but are used in performance evaluation of many other areas. For instance, in evaluating the speed of a computer, one generally defines a reference computer, such as the VAX 11/780, and computes the speedup of the computer with respect to the reference for a collection of benchmarks. Based on the evaluation results, one generalizes the speedup to benchmarks not tested in the evaluation.

Since different regions of the problem space of an application domain may have different characteristics, it may not be possible to evaluate generalization across all examples of a problem space. To this end, the problem space is decomposed into smaller partitions before generalization is evaluated. For instance, in evaluating a computer, one defines its speedups for different collections of benchmarks in order to reflect its performance under different applications.

In the partitioning of a problem space, we define a *problem subspace* as a user-defined partition of a problem space so that hypotheses for one subspace are evaluated independent of hypotheses in other subspaces. Such partitioning is generally guided by common-sense knowledge or by user experience in solving similar application problems. To identify a problem subspace, we need to know one or more attributes to classify test cases and a set of decision rules to identify the subspace to which a test case belongs. For instance, in evaluating the speedup of a computer, the partitioning of the class of all applications is guided by user experience into the class of scientific applications and the class of business applications.

Given a subspace of test cases, we define a *problem subdomain* as a partitioning of the subspace into smaller partitions so that the evaluation of a hypothesis can be done quantitatively for all the test cases in a subdomain. Such partitioning is necessary because the statistical performance metrics computed (such as average or maximum) is not meaningful when the performance values are of different ranges and distributions. To continue from the previous example, the class of scientific benchmarks are further partitioned into subdomains according to their computational behavior, such as whether a program is CPU-bound (central processing unit-bound) or I/O-bound (input/output-bound).

In the same way that test cases are partitioned into subspaces, we need to know the attributes to classify test cases and a set of decision rules to identify the subdomain to which a test case belongs. This may be difficult in some applications because the available attributes may not be well defined or may be too large to be useful. For instance, the attribute to classify whether a benchmark program is CPU-bound or I/O-bound is imprecise and may depend on many underlying characteristics of the program.

After evaluating the performance of a hypothesis in each subdomain, we need to compare its performance across subdomains. For instance, one would be interested to know whether a computer has high speedups across both CPU-bound and I/O-bound applications. This comparison may be difficult because test cases in different subdomains of a subspace may have different performance distributions and cannot be compared statistically. We address this issue in a later section. In the next subsection, we examine some formal results in generalizability for classification problems in one subdomain.

Formal Results on Generalizability

Formal methods to deal with generalizability in learning with one performance measure have been studied extensively in computational learning theory. They center on the notion of PAC-learnability (23) of a concept class **C** by a learning algorithm L, where a concept is defined as a subset of some instance space X. A learner tries to learn target concept C, finding out points of X (drawn randomly) whether they belong to the target concept. The goal of the learner is to produce with high probability ($>1 - \delta$) a hypothesis that is close (within ϵ) to the target concept, assuming that the learner does not know the underlying distribution of the sample points. (The following definitions are from a survey paper by Kearns et al. (24).) A concept C produced by a learning algorithm L on input vector T is *approximately correct* if the error rate $P(C \oplus T)$ is at most ϵ. If, for any concept class **C**, with probability distribution P, accuracy parameter ϵ, and confidence parameter δ, the probability that the output C is approximately correct is at least $(1 - \delta)$, then the learning algorithm is *probably approximately correct;* and, L is said to PAC-learn **C**. A learning algorithm L is a polynomial PAC-learning algorithm for class **C**, if L PAC-learns **C** with both time complexity and sample complexity polynomial in $1/\epsilon$ and $1/\delta$.

To understand bounds on estimation by a learning algorithm, we need to estimate the largest number of input-space points for which almost every possible dichotomy is achieved by some concept from a class **C**. VC-dimension (named after Vapnik and Chervonenkis (25)) addresses this issue. VC-dimension, V, of a concept class **C** is the size of the largest set **S** of input-space points such that for every subset $\mathbf{U} \subseteq \mathbf{S}$, there exists some concept $C \in \mathbf{C}$ where $\mathbf{U} = \mathbf{S} \cap \mathbf{C}$. C is some function realized by the concept; and **C**, the set of all such functions realizable by that concept.

Sauer (26) notes that whenever the VC-dimension of a function class is finite, the number of dichotomies grows subexponentially (actually, polynomially) in the number of points. The probability of a concept learned with a large estimation error producing correct outputs for a given set of points goes rapidly to zero as the size of the set increases. A learning algorithm whose outputs are always consistent with the examples seen so far is called a *consistent PAC-learning algorithm.* If the VC-dimension of a concept class is finite, then a consistent learning algorithm trained on a sufficiently large set of examples is likely to learn the correct concept.

Blumer et al. (27) have derived bounds on the number $m(\epsilon, \delta)$ of examples needed by a consistent algorithm to PAC-learn a concept class **C** having VC-dimension d. This was improved by Ehrenfeucht et al. (28) to $(1/\epsilon \ln 1/\delta + d/\epsilon)$.

Baum and Haussler (29) have used these results to relate the size of a neural network, the accuracy of the learned concept, and the number of examples needed in order to guarantee a particular degree of accuracy. Their analysis suggests

Table 1. Summary of Raw CPU Times of Four Computers in Evaluating Three Benchmarks

	Computer			
Benchmark	C_{75}	C_{76}	C_{86}	C_{99}
t_1	30.19	30.31	26.21	40.61
t_2	43.12	43.34	34.09	24.65
t_3	71.93	72.49	104.51	98.41

that generalization can be improved by pruning unnecessary hidden units during learning. The reduced architecture has VC-dimension not significantly larger than the VC-dimension for an optimal number of hidden units. Baum and Haussler establish the following necessary and sufficient conditions for valid generalization for learning in neural networks of thresholded binary units.

- A network of N nodes and W weights, which after being trained on at least $O(W/\epsilon \log N/\epsilon)$ examples, classifies at least $(1 - \epsilon/2)$ of them correctly, will almost certainly classify a fraction $(1 - \epsilon)$ of future examples correctly.
- A fully connected feedforward network with one hidden layer, trained on fewer than (W/ϵ) examples will, for a dichotomy realizable by the network, fail to find the requisite set of weights for more than a fraction $(1 - \epsilon)$ of future examples.

Haussler (30) shows that, for it to be likely that feedforward networks with sigmoidal units obtain a low estimation error, the number of examples must grow linearly with both the number of modifiable weights and the number of hidden layers. That is, either of the following desiderata demands a larger training sample: (1) lowering the estimation error; (2) increasing the confidence; and, (3) learning with sigmoids having a higher slope.

Barron (31) shows that, for a feedforward network having n sigmoidal units and d input units and trained on N examples, the total mean squared error (approximation plus estimation) between the true function and the estimated function is bounded from above by $O(1/n) + O(nd/N) \log N$.

In summary, the theory in learnability provides conditions and bounds on generalization that are useful when certain restricted assumptions are met. Such assumptions may be difficult to ascertain in practice because it is difficult to characterize the set of test cases and hypotheses precisely. Under such conditions, heuristic methods to measure generalizability need to be developed. In the next two subsection, we present some results in this area.

Anomalies in Performance Normalization

In general learning problems, the raw performance results obtained in evaluating hypotheses on examples may depend on the size and characteristics of the examples and may not be directly comparable. For instance, Table 1 shows the CPU times of four computers in evaluating three benchmarks. Obviously, these performance values cannot be aggregated directly because they belong to different ranges and are of different distributions. To aggregate them statistically, we must normalize them first. In the following, we show five different normalization methods.

Average Improvement Ratio. Using the performance values of one hypothesis as the baseline, we normalize each performance value of another hypothesis by computing its ratio with respect to that of the baseline when tested on the same example. The average of the improvement ratios is then used as the aggregate performance measure. The drawback of this approach is that different ordering of the hypotheses can be obtained, depending on the baseline hypothesis used. To illustrate this point, consider the performance data presented in Table 1. The second column of Table 2 shows the three anomalous orderings of the four computers based on their average normalized speedups using each computer as the baseline for normalization. This shows that generalization based on the average improvement ratios does not always lead to consistent conclusions.

Average Symmetric Improvement Ratio. This is a normalization method we have developed before to avoid anomalies in inconsistent orderings of two hypotheses due to the choice of the baseline hypothesis (32). The idea is to avoid emphasizing differently in different ranges of the normalized performance values. The symmetric improvement ratio is defined as follows:

$$S_{\text{sym}+,i} = \begin{cases} S_{+,i} - 1 & \text{if } S_{+,i} \geq 1 \\ 1 - \dfrac{1}{S_{+,i}} & \text{if } 0 \leq S_{+,i} < 1 \end{cases} \tag{1}$$

$$\overline{S}_{\text{sym}+} = \frac{1}{m} \sum_{i=1}^{m} S_{\text{sym}+,i} \tag{2}$$

where $S_{+,i}$ is the original improvement ratio on the ith test case. The symmetric improvement ratio has the property that improvements are in the range between 0 and infinity, and degradations are in the range between 0 and negative infinity. For two hypotheses, when we reverse the role of the baseline hypotheses, their symmetric improvement ratios only change in sign. Hence, symmetric improvement ratios avoid anomalies in performance orderings with two hypotheses.

However, anomalies in performance ordering are still present when more than two hypotheses are concerned. This is illustrated in Table 2 that shows three different orderings when different computers are used as the baseline. Hence, generalization based on the average symmetric improvement ratios may not lead to consistent conclusions.

Harmonic Mean Performance. This is defined as follows:

$$\overline{S_h} = \frac{m}{\sum_{i=1}^{m} 1/S_+} \tag{3}$$

Again, as illustrated in Table 2, anomalies in orderings are still present.

Table 2. Anomalous Orderings of Computers in Decreasing Average Normalized Speedups Using Three Different Normalization Methods

Baseline	Average Improvement Ratio	Average Symmetric Improvement Ratio	Harmonic Mean
C_{75}	$C_{99}C_{86}C_{75}C_{76}$	$C_{99}C_{75}C_{76}C_{86}$	$C_{75}C_{76}C_{86}C_{99}$
C_{76}	$C_{99}C_{86}C_{75}C_{76}$	$C_{99}C_{75}C_{76}C_{86}$	$C_{75}C_{76}C_{86}C_{99}$
C_{86}	$C_{75}C_{76}C_{99}C_{86}$	$C_{75}C_{76}C_{85}C_{99}$	$C_{86}C_{75}C_{76}C_{99}$
C_{99}	$C_{75}C_{76}C_{86}C_{99}$	$C_{86}C_{99}C_{75}C_{76}$	$C_{99}C_{86}C_{75}C_{76}$

Geometric Mean Performance. This is defined as follows:

$$\overline{S_g} = \sqrt[m]{\prod_{i=1}^{m} S_{+,i}} \tag{4}$$

Taking the logarithm of both sides, we have:

$$\log \overline{S_g} = \frac{1}{m}\sum_{i=1}^{m} \log S_{+,i} = \frac{1}{m}\sum_{k=1}^{m} \log t_{b,k} - \frac{1}{m}\sum_{k=1}^{m} \log t_{h,k} \tag{5}$$

where $t_{b,k}$ and $t_{h,k}$ are, respectively, the kth performance values of the baseline and the hypothesis being normalized. Based on Eq. (5), an alternative way to view a geometric mean is that it is an arithmetic mean of the logarithms of raw performance values. The effect of the baseline hypothesis on the average normalized performance is reflected in the first constant term in Eq. (5). Hence, when the baseline is changed, only a constant term will be changed, and performance ordering is not affected. This is illustrated in the example in Table 1 in which the ordering $C_{86}C_{75}C_{76}C_{99}$ is unchanged when the baseline is changed.

Average Normalized Performance With Respect to the Median Performance. This belongs to a general class of methods that normalizes the performance values of hypotheses on each test case with respect to a test case-specific constant that is invariant as more hypotheses are evaluated. The specific method here uses the median performance value of all the hypotheses on each test case as the baseline for normalization. Unlike using a baseline hypothesis that may induce a different ordering when the baseline is changed, the median performance is invariant with respect to the hypotheses and test cases in a subdomain. Using this normalization method, the performance distributions of all the test cases will center around zero. This method is illustrated in the example in Table 1 in which the ordering is $C_{76}C_{75}C_{99}C_{86}$. In computing this ordering, we made a simplifying assumption that the median performance of each computer on the three benchmarks is the same as the median performance of the computer across all possible benchmarks.

A potential problem with this approach is the unavailability of the true median performance value of hypotheses for each test case. Hence, the sample median may have to be used instead. Unfortunately, estimated sample medians are inaccurate during learning because hypotheses may not be tested adequately, and sample medians are sensitive to the hypotheses tested. Solutions to this issue are still open at this time.

In summary, anomalies in performance normalization do not exist when either the baseline is fixed (as in the case of the median performance) or the effect of changing the baseline only results in changing a constant term in the (transformed) normalized performance (as in the case of the geometric mean performance). In other cases, it is possible for the order of the hypotheses to change when the baseline is changed. The necessary and sufficient conditions for anomalies to happen are still open at this time.

Generalizability Measures Across Subdomains

When hypotheses are tested across different subdomains of an application, their performance values, even after normalization, may have different ranges and different distributions. As a result, these performance values cannot be aggregated statistically, and the hypotheses cannot be compared directly and generalized across subdomains. In this section, we present a heuristic method to evaluate performance across subdomains in a range-independent way. We assume that the performance values of testing a hypothesis in a subdomain are independent and identically distributed. This assumption allows the values in a subdomain to be aggregated by statistical methods, such as averaging.

In the following, we present a method that uses the sample mean as a statistical estimate of the population mean. To address uncertainties in using sample means, we have studied a concept called probability of win (32), P_{win}, that compares two sample means and computes the probability that one sample mean is larger than another. This is similar to hypothesis testing in which we take random samples to test whether a property of a population is likely to be true or false (33). Obviously, it may be difficult to test a hypothesis fully by testing the entire population of test cases or by testing only a single random sample.

There are four steps in general hypothesis testing. (1) Specify a significance level α. (2) Specify the testing hypotheses that include both null hypothesis H_0 and alternative hypothesis H_1. (3) Find the corresponding acceptance region using lookup tables. (4) Make a decision on the sample value. If the sample falls in the acceptance region, then accept H_0 and reject H_1; otherwise, reject H_0 and accept H_1.

The probability of win measures statistically how much better (or worse) the sample mean of one hypothesis is as compared to that of another. It resembles the significance level in general hypothesis testing, but there are two major differences. First, only one hypothesis $\{H: \mu 1 > \mu 2\}$ is specified, without the alternative hypothesis. Further, in contrast to hypothesis testing, acceptance confidence is not given in advance but is evaluated based on sample values.

One advantage of P_{win} is that it is between zero and one and is independent of the actual performance difference across subdomains. Hence, it can be used to compare hypotheses in a uniform way across subdomains.

Consider the performance of H_i in subdomain j. (For convenience of formulation, subscript j is ignored in the following discussion.) Let μ_i and σ_i be the true mean and true standard deviation of the mean normalized performance with respect to the baseline hypothesis H_0 (any one of the normalization methods presented in the previous section can be used). When n_i samples are taken, we can calculate the sample mean $\overline{\mu}_i$ and sample standard deviation $\overline{\sigma}_i$. By Central Limit Theorem,

$$Pr(\bar{\mu}_i \mid \mu_i, \sigma_i, n_i) \approx \mathcal{N}\left(\mu_i, \frac{\sigma_i^2}{n_i}\right)$$

where $\mathcal{N}$ is the normal distribution function with mean μ_i and standard deviation $\sqrt{\sigma_i^2/n_i}$. Let t be

$$t = \frac{\bar{\mu}_i - \mu_i}{\bar{\sigma}_i / n_i}$$

where t has Student's t-distribution with $n_i - 1$ degrees of freedom when the number of samples is less than 30 and the variance is unknown. The probability that this hypothesis is better than H_0 with mean value zero (if the average normal-

Table 3. Examples Illustrating How P_{win} Changes With Increasing Number of Samples (Performance is Normalized With Respect to H_0)

			No. of Samples to Compute P_{win}		
H_i	μ_i	σ_i	5	10	30
H_1	0.336	1.231	0.652	0.725	0.849
H_2	−0.129	0.222	0.202	0.097	0.012
H_3	0.514	0.456	0.940	0.991	1.000

ized performance of H_0 is not zero, then appropriate shifting in the mean value to zero can be performed) is

$$Pr(H\ is\ true) = Pr\left(t \in \left(-\infty, \frac{\bar{\mu}_i}{\bar{\sigma}_i/\sqrt{n}}\right)\right) \tag{6}$$

$$= \int_{-\infty}^{\bar{\mu}_i/(\bar{\sigma}_i/\sqrt{n})} p(t\ is\ t\text{−}distributed)\, dt \tag{7}$$

where the acceptance region of this hypothesis is $(-\infty, \overline{\mu}_i/(\overline{\sigma}_i/\sqrt{n})$. Note that the right bound of the acceptance region is a random variable that depends on both the sample mean and the sample variance.

Example 1. Table 3 illustrates the P_{win} for three hypotheses. We see that P_{win} of H_1 increases toward one when the number of samples increases. (H_1 is better than H_0.) In contrast, P_{win} of H_2 reduces to zero when the number of samples is increased. (H_2 is worse than H_0.) Last, P_{win} of H_3 reaches the maximum value 1.0, which means H_3 is definitely better than H_0.

Note that P_{win} considers both the mean and variance. Hence, when P_{win} of a hypothesis is close to 0.5, it is not clear whether the hypothesis is better than or worse than the baseline.

Given baseline hypothesis H_0, we now show P_{win} of H_i in subdomain j with respect to the average performance of H_0. Assuming sample mean $\hat{\mu}_{i,j}$, sample variance $\hat{\sigma}^2_{i,j}$, and $n_{i,j}$ test cases, P_{win} is defined as follows:

$$P_{\text{win}}(i, j) = F_{\text{t}}\left(n_{i,j} - 1, \frac{\hat{\mu}_{i,j}}{\sqrt{\hat{\sigma}^2_{i,j}/n_{i,j}}}\right) \tag{8}$$

where $F_t(\nu, x)$ is the cumulative distribution function of Student's t-distribution with ν degrees of freedom, and $P_{win}(i, j)$ is the probability that the true performance (population mean) of H_i in subdomain j is better than that of H_0. When $n_{i,j} \rightarrow \infty$, we have

$$P_{\text{win}}(i, j) \approx \Phi\left(\frac{\hat{\mu}_{i,j}}{\sqrt{\hat{\sigma}^2_{0,j}/n_{i,j}}}\right) \tag{9}$$

where Φ is the standard cumulative normal distribution function (34).

It is important to point out that probabilities of mean are used to evaluate whether a hypothesis is better than the baseline and is not meant to rank order all the hypotheses. Hence, when hypotheses are ordered using their probabilities of win and performance is normalized by any method in which the baseline can be changed, anomalies in performance ordering may happen. As illustrated in the Ref. 35, this phenomenon happens because not only the mean but the variance of the baseline are important in determining the ordering of the hypotheses. The variance of the performance values places another degree of freedom in the performance ordering, which can change the ordering when the variance changes. Consequently, the ordering may change when a baseline with a small variance is changed to one with a large variance (or vice versa). This happens even for normalization methods that do not have anomalies when hypotheses are ordered by their mean values, such as the geometric mean. In short, anomalies in performance ordering will not happen when hypotheses are ranked by their probabilities of mean and when the baseline hypothesis is fixed, such as the case when the median performance of the hypotheses is used as the baseline.

We are now ready to define a generalizability measure across multiple subdomains. Because different subdomains have different statistical behavior, performance from different subdomains must be treated independently and cannot be combined.

There are two assumptions on the strategies presented here.

- We assume that the set of subdomains used in the design process are representatives of all the subdomains in the application. These subdomains behave in a statistically similar fashion to subdomains used in learning and in generalization.
- We assume that the relative importance of one subdomain as compared to another is unknown, and that the performance of hypotheses in subdomains may be dependent. Under these assumptions, we cannot aggregate performance values of hypotheses across subdomains. Our strategy is to select hypotheses so that their worst-case performance across all subdomains is better than a minimum level.

The objective of generalization here is to select a hypothesis that is better than the incumbent hypothesis over a problem domain. When there are multiple such hypotheses, our procedure should attempt to maximize the likelihood of selecting the best hypothesis among the given set. Define:

$$P_{\text{WIN}}(i) = min_j P_{\text{win}}(i, j) \tag{10}$$

When there is a baseline hypothesis H_0, we apply one of the strategies in a previous section to normalize the performance of a hypothesis in a subdomain with respect to the baseline hypothesis. We consider H_i to be better than H_0 in subdomain j when $P_{WIN}(i) > 0.5 + \Delta$. Note that $P_{WIN}(i)$ is independent of subdomain j and can be used in generalization if it were true across all subdomains, even those subdomains that were not tested in learning.

The following are three possible outcomes when comparing $P_{WIN}(i)$ of H_i to H_0.

1. H_i is the only hypothesis that is better than H_0 in all subdomains. H_i can then be chosen as the hypothesis for generalization.
2. Multiple hypotheses are better than H_0 in all subdomains. Here, we should select one hypothesis that maximizes the likelihood of being better than H_0 over the entire domain. This likelihood (or degree of confidence) can be adjusted by increasing Δ, which is equivalent to placing a tighter constraint in each subdomain, hence eliminating some potential hypotheses that are found to be better than H_0 under a looser constraint.
3. No hypothesis is better than H_0 in all subdomains. Since no hypothesis is superior to H_0, H_0 is the most generalizable.

Alternatively, it is possible to find hypotheses such that $P_{\text{WIN}} > 0.5 + \Delta$ where $\Delta < 0$. Such hypotheses have less certainty in performing better than H_0 across all the subdomains. However, since P_{WIN} is based on the worst-case P_{win} across all the subdomains, hypotheses selected this way may still perform better than the baseline in some subdomains. Such hypotheses should be considered as alternatives to H_0.

We have considered so far generalization based on one performance measure. In general, there may be multiple performance measures in an application, and generalization determines whether a hypothesis behaves consistently across all subdomains with respect to all the performance measures. The problem belongs to a general class of multi-objective optimization problems that can be solved in some special forms. In our approach, we propose to constrain all but one measures and to optimize the unconstrained measure subject to the constraints (32). The constraints in this approach are defined with respect to the performance of an existing baseline hypothesis. This is similar to first normalizing the performance with respect to that of the baseline and formulating a constraint such that the normalized performance is larger than one. Again, care must be taken in normalization because anomalies in performance ordering may happen when certain normalization methods are used and the baseline is changed.

Probabilities of mean have been used to evaluate generalizability in various genetics-based learning and generalization experiments (17,32,33,35,36,37,38,39). These include the learning of load balancing strategies in distributed systems and multicomputers, the tuning of parameters in very large scale integration (VLSI) cell placement and routing, the tuning of fitness functions in genetics-based VLSI circuit testing, the automated design of feedforward neural networks, the design of heuristics in branch-and-bound search, range estimation in stereo vision, and the learning of parameters for blind equalization in signal processing.

SUMMARY

In this article, we have defined the generalization problem, summarized various approaches in generalization, identified the credit assignment problem, and presented some solutions in measuring generalizability.

Existing formal results in measuring generalizability only address some restricted cases in which performance measures are from one common (possibly unknown) distribution. In general, the performance of applications may be measured by multiple metrics, and test cases may be grouped into subsets (or subdomains) such that each subset has a different performance distribution. Consequently, existing methods cannot be used to measure generalizability across subdomains of test cases.

We have presented some systematic methods to evaluate generalizability within a subdomain. To eliminate dependence on the size of a test case in a subdomain, we have shown various normalization methods to normalize performance with respect to a baseline hypothesis. Some of these methods can lead to anomalies in orderings when hypotheses are rank-ordered by the average normalized measure and the baseline is changed. Only when the baseline hypothesis is fixed (like using the median performance as the baseline) or when the effect of the baseline only exists as a constant in the average normalized measure (like using the geometric mean) can anomalies be eliminated.

Finally, we have presented some methods to evaluate generalizability across subdomains. We have introduced a concept called probability of mean that measures the probability that the sample mean of a hypothesis is better than the population mean of the baseline, given the number of samples tested and the variance of the samples. As probabilities of win are in the range between zero and one, they can be used to evaluate generalizability across subdomains. Unfortunately, probabilities of win cannot be used to rank-ordered hypotheses, even when performance is normalized and averaged using methods like the geometric mean. This happens because probabilities of mean are used to evaluate whether a hypothesis is better than the baseline and is not meant to rank order all the hypotheses. The variance of the performance values places another degree of freedom in the ordering, leading to a different ordering when a baseline with a different variance is used.

ACKNOWLEDGMENT

Research supported by National Science Foundation Grant MIP 96-32316 and National Aeronautics and Space Administration Grant NAG 1-613.

BIBLIOGRAPHY

1. *Encyclopaedia Britannica CD'95,* Encyclopaedia Britannica, Inc., 1995.
2. A. Barr and E. A. Feigenbaum, *The Handbook of Artificial Intelligence,* volume 1, 2, and 3, Los Altos, CA: William Kaufmann, 1981.
3. T. M. Mitchell, Generalization as search, *Artificial Intelligence,* **18**: 203–226, 1982.
4. D. A. Waterman, Generalization learning techniques for automating the learning of heuristics, *Artificial Intelligence,* **1** (1/2): 121–170, 1970.
5. N. J. Nilsson, *The Mathematical Foundations of Learning Machines,* Palo Alto, CA: Morgan Kaufman, 1990.
6. J. R. Koza, *Genetic Programming,* Cambridge, MA: The MIT Press, 1992.
7. J. H. Holland, *Adaptation in Natural and Artificial Systems,* Ann Arbor, MI: Univ. of Michigan Press, 1975.
8. L. B. Booker, D. E. Goldberg, and J. H. Holland, Classifier systems and genetic algorithms, *Artificial Intelligence,* **40** (1): 235–282, 1989.

9. J. L. McClelland and D. E. Rumelhart, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations,* Cambridge, MA: Bradford Books (MIT Press), 1985.
10. G. F. DeJong and R. J. Mooney, Explanation-based learning: an alternative view, *Machine Learning,* **1** (2): 145–176, 1986.
11. T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli, Explanation-based generalization: a unifying view, *Machine Learning,* **1** (1): 47–80, 1986.
12. P. Mehra and B. W. Wah (eds.), *Artificial Neural Networks: Concepts and Theory,* Los Alamitos, CA: IEEE Computer Society Press, 1992.
13. M. Kearns et al., Recent results on Boolean concept learning, in *Proc. Fourth Int'l. Workshop on Machine Learning,* Palo Alto, CA: Morgan Kaufmann, 1987.
14. H. A. Simon and G. Lea, Problem solving and rule induction: a unified view, in L. W. Gregg (ed.), *Knowledge and Cognition,* pages 105–127. Potomac, MD: Erlbaum, 1974.
15. R. S. Sutton, *Temporal Credit Assignment in Reinforcement Learning,* Ph.D. thesis, Univ. of Massachusetts, Amherst, MA, February 1984.
16. D. Nguyen and B. Widrow, The truck backer-upper: An example of self-learning in neural networks, in *Proc. Int'l Joint Conf. on Neural Networks,* volume II, pages 357–363. IEEE, 1989.
17. C.-C. Teng and B. W. Wah, Automated learning of the minimal configuration of a feed forward neural network, *IEEE Trans. Neural Netw.,* **7**: 1072–1085, 1996.
18. W. W. Cohen, Generalizing number and learning from multiple examples in explanation based learning, *Machine Learning,* pages 256–269, 1988.
19. B. W. Wah, Population-based learning: a new method for learning from examples under resource constraints, *IEEE Trans. Knowl. Data Eng.,* **4**: 454–474, 1992.
20. S. F. Smith, Flexible learning of problem solving heuristics through adaptive search, in *Proc. Int'l Joint Conf. on Artificial Intelligence,* pages 422–5. Morgan Kaufman, 1983.
21. J. H. Holland, Properties of the bucket brigade algorithm, in J. J. Grefenstette (ed.), *Proc. Int'l. Conf. Genetic Algorithms and Their Applications,* pages 1–7, The Robotics Inst. of Carnegie-Mellon Univ., Pittsburgh, PA, 1985.
22. J. J. Grefenstette, Credit assignment in rule discovery systems based on genetic algorithms, *Machine Learning,* **3** (2/3): 225–246, October 1988.
23. L. G. Valiant, A theory of the learnable, *Commun. ACM,* **27** (11): 1134–1142, November 1984.
24. M. Kearns et al., Recent results on Boolean concept learning, *Machine Learning,* pages 337–352, 1987.
25. V. N. Vapnik and A. Ya. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, *Theoretical Probability and its Applications,* **XVI** (2): 264–280, 1971.
26. N. Sauer, On the density of families of sets, *J. Combinatorial Theory (A),* **13**: 145–147, 1972.
27. A. Blumer et al., Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proc. 18th Symp. on Theory of Computing,* pages 273–282. ACM, 1986.
28. A. Ehrenfeucht et al., A general lower bound on the number of examples needed for learning. In D. Haussler and L. Pitt (eds.), *Proc. 1988 Workshop on Computational Learning Theory,* pages 139–154, Palo Alto, CA, Morgan Kaufmann, 1988.
29. E. B. Baum and D. Haussler, What size net gives valid generalization? In D. Z. Anderson (ed.), *Proc. Neural Information Processing Systems,* pages 81–90, New York, American Inst. of Physics, 1988.
30. D. Haussler, Generalizing the PAC model: Sample size bounds from metric dimension-based uniform convergence results, in *Proc. 30th Annu. Symp. Foundations Comput. Sci.,* pages 40–45. Computer Society Press, IEEE, 1989.
31. A. R. Barron, Approximation and estimation bounds for artificial neural networks, in *Proc. 4th Annual Workshop on Computational Learning Theory (COLT'91),* pages 243–249, Palo Alto, CA, Morgan Kaufmann, 1991.
32. B. W. Wah et al., Genetics-based learning of new heuristics: rational scheduling of experiments and generalization, *IEEE Trans. Knowl. Data Eng.,* **7**: 763–785, 1995.
33. A. Ieumwananonthachai and B. W. Wah, Statistical generalization of performance-related heuristics for knowledge-lean applications, *Int. J. Artificial Intelligence Tools,* **5** (1 2): 61–79, June 1996.
34. J. L. Devore, *Probability and Statistics for Engineering and the Sciences,* Monterey, CA: Brooks/Cole Pub. Co., 1982.
35. B. W. Wah, A. Ieumwananonthachai, and T. Yu, Genetics-based learning and statistical generalization, in S. Tzafestas (ed.), *Knowledge-Based Systems: Advanced Concepts, Tools and Applications,* pages 319–347, Singapore: World Scientific Publishing Co., 1997.
36. P. Mehra and B. W. Wah, A systematic method for automated learning of load-balancing strategies in distributed systems, *Int. J. Syst. Sci.,* (accepted to appear) 1997.
37. B. W. Wah, A. Ieumwananonthachai, and Y. C. Li, Generalizationi of heuristics learned in genetics based learning, in S. K. Pal and P. Wang (eds.), *Genetic Algorithms and Pattern Recognition,* pages 87–126, Boca Raton, FL: CRC Press, 1996.
38. A. Ieumwananonthachai and B. W. Wah, Statistical generalization of performance-related heuristics for knowledge-lean applications, in D. Dasgupta and Z. Michalewicz (eds.), *Evolutionary Algorithms in Engineering Applications,* pages 293–313. New York, NY: Springer Verlag, 1997.
39. B. W. Wah and A. Ieumwananonthachai, Generalization of heuristics learned in genetics based learning with applications in circuit testing and computer aided design, in Xin Yao (ed.), *Evolutionary Computation.* Singapore: World Scientific Publishing Co. Pte. Ltd., 1996.

Benjamin W. Wah
University of Illinois, Urbana-Champaign

ARTIFICIAL INTELLIGENCE PLANNING.

See Planning.