A Randomized ANOVA Procedure for Comparing Performance Curves

Justus H. Piater
piater@cs.umass.edu

Paul R. Cohen
cohen@cs.umass.edu

Xiaoqin Zhang
xqzhang@cs.umass.edu

Michael Atighetchi
adi@cs.umass.edu

Computer Science Department
University of Massachusetts
Amherst, MA 01003

Abstract

Three factors are related in analyses of performance curves such as learning curves: the amount of training, the learning algorithm, and performance. Often we want to know whether the algorithm affects performance and whether the effect of training on performance depends on the algorithm. Analysis of variance would be an ideal technique but for carryover effects, which violate the assumptions of parametric analysis of variance and can produce dramatic increases in Type I errors. We propose a novel, randomized version of the two-way analysis of variance which avoids this problem. In experiments we analyze Type I errors and the power of our technique, using common machine learning datasets.

1 INTRODUCTION

A common task in machine learning is comparative assessment of learning methods. Most research on this issue focuses on performance measures such as classification accuracy after training, or percentage of games won by a game-playing program (e.g. Mitchell 1997 ch. 5, Dietterich (1998), Rasmussen et al. 1996). However, it is sometimes interesting to compare time series of performance, such as learning curves. For example, two algorithms might have comparable asymptotic performance, but we would like to test the hypothesis that one achieves this level of performance more quickly than the other.

Which statistical procedures are appropriate to identify differences between the performance of algorithms over time, and particularly during training? One obvious approach might be to apply the aforementioned methods repeatedly at different times, comparing the performance of algorithms at each of several levels of training. Unfortunately, multiple comparisons can lead to overestimates of the significance of results (see Section 2) and are inappropriate for comparing performance curves.

A better approach is to describe differences between algorithms during training in terms of two effects:

Algorithm Effect: Does one algorithm generally achieve higher performance than another?

Interaction Effect: Does the influence of training on performance depend on the algorithm?

Figures 1a and 1b illustrate prototypical cases for each effect. In practice, however, some combination of effects will occur. In Figure 1c, for instance, both curves start out with similar slopes, but one of them converges to a lower asymptote. Figure 1d shows a case where both curves start at the same point and achieve similar asymptotic performances, but one algorithms learns faster (with respect to the amount of training) than the other. In this latter case, we find that both algorithm and interaction effects concentrate in the early stages of training, and both effects essentially disappear with increasing amount of training.

This paper presents a method for detecting Algorithm and Interaction effects in learning curves. Actually, the method is not restricted to learning curves, it applies to any kind of performance curves. The method tests two hypotheses:

- The mean performances of two or more algorithms are the same (no Algorithm effect).
- The relationship between training and performance does not depend on Algorithm (no Interaction effect).

Such effects are typically tested with *analysis of variance* (ANOVA). However, the conventional parametric ANOVA is

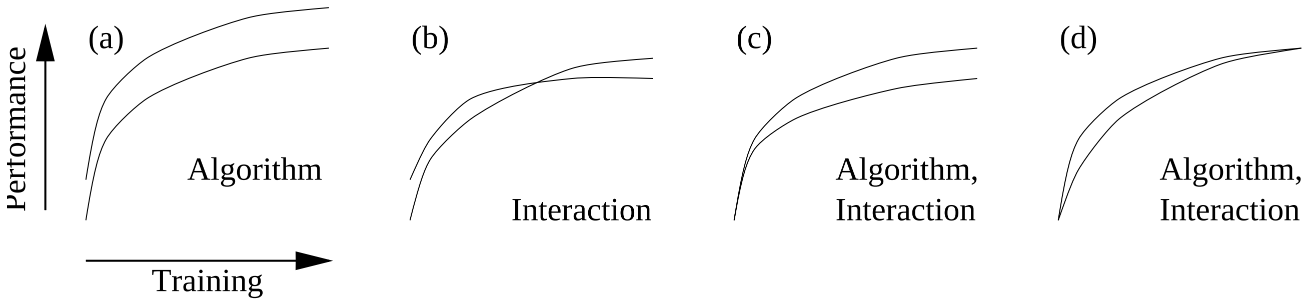

Figure 1: Some kinds of differences between learning curves. The statistical effects on performance (Algorithm and/or Interaction effects) are listed for each situation. In case *c*, the Interaction effect disappears at the later stages of training; in case *d*, both effects disappear.

based on several assumptions, of which one, homogeneity of covariance, is strongly violated by most time series data. In particular, conventional ANOVAs on learning curves can dramatically overestimate the significance of algorithm effects and underestimate the significance of interaction effects. Following some statistical preliminaries in Section 2, we demonstrate how ANOVA gives incorrect results for learning curves (Section 3) and then introduce our novel procedure, a randomized version of ANOVA (Section 4). The remainder of the paper presents experimental results with conventional and randomized ANOVA, comparing the power and Type I errors of the methods.

2 STATISTICAL HYPOTHESIS TESTING

This section defines terms and may safely be skipped by readers familiar with statistical hypothesis testing.

Hypothesis testing involves these steps: Assert a *null hypothesis* H_0. Decide on a statistic ϕ. Collect a sample s of size n and calculate $\phi(s)$ for the sample. Derive the probability distribution $\mathcal{S}$ of all possible values of $\phi(i)$ for samples i of size n *under* H_0. These restrictions are important: $\mathcal{S}$ isn't the distribution of ϕ for *any* sample, but for samples of size n that would arise if the null hypothesis were true. $\mathcal{S}$ is called the *sampling distribution* of ϕ. One may then ask, "What is the probability of obtaining a statistic value of $\phi(s)$ or more by chance if H_0 were true?" The answer, called a p value, is the area of $\mathcal{S}$ above $\phi(s)$. Suppose $p = .01$. Should you reject the null hypothesis? There isn't a correct answer to this question, but you can be assured that if you do reject H_0, the probability that you do so in error is no greater than p. Rejecting H_0 when it is true is called a *Type I error*. Failing to reject H_0 when it is false is a Type II error, and the *power* of a test—the probability that you will reject H_0 when it is false—is one minus the probability of a Type II error.

One may also ask, "What value of $\phi(s)$ must I exceed to be assured that my p value is less than some threshold α?" This is called the *critical value* of ϕ and, obviously, it varies with α.

One should not compare performance curves by repeatedly comparing points on the curves (e.g., comparing performance after $i, 2i, 3i \ldots$ training instances). Each comparison will with some probability α assert a difference in performance when in reality there is none — a Type I error. If the comparison procedure is applied m times, to m pairs of points on learning curves, then the *total* probability of Type I error is roughly $1 - (1 - \alpha)^m$. (The probability is exactly $1 - (1 - \alpha)^m$ if the comparisons are independent, but they are not, and their non-independence necessitates the technique developed in this paper.) One can control the total probability of a Type I error, but only by reducing α — which increases the critical values for individual comparisons — making it less likely that comparisons will find differences that actually exist. Said differently, the power of the tests is reduced (see Cohen 1995 for a discussion of related issues). Multiple comparisons are not the right tool for comparing performance curves.

3 ANOVA FOR COMPARING PERFORMANCE CURVES

Suppose we have two learning algorithms A_1 and A_2, each of which trains l times on a set of k instances, e.g., in an l-fold cross validation procedure. Then we have l estimates of the performance of each algorithm at each level of training. Put another way, we have l "lines" $L_1^{(1)}, \ldots, L_l^{(1)}$ for A_1 and another l lines $L_1^{(2)}, \ldots, L_l^{(2)}$, where each line is a list of k numbers that represent the performance of the algorithm at level h ($1 \leq h \leq k$) of training, on that particular fold of the cross validation. A schematic data table is shown in Figure 2, where the axes of the table represent the factors *Training* and *Algorithm*. Lines may of course be generated by methods other than cross-validation; for example, they might represent training on several different datasets. The important thing is that the data points on a line are not independent. In statistical parlance, they are *repeated measures* and they create *carryover effects*, meaning that the performance represented by earlier points on a line influences, or carries over to, later performance.

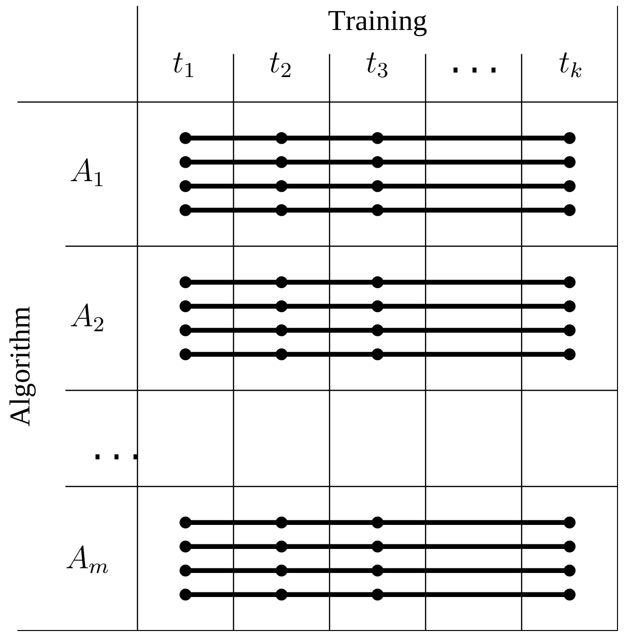

Figure 2: Data table setup for randomized ANOVA. This example shows $l = 4$ learning curves per algorithm.

Were it not for these carryover effects, analysis of variance would be an ideal tool to analyze learning curves. Analysis of variance tests for *main effects* of factors and *interaction effects* between factors. Each kind of effect is represented by an F statistic, which has an expected value of 1.0 under the null hypothesis of no effect. Formulae for calculating F are straightforward and widely available (e.g., see Cohen 1995) and will not be repeated here. The patterns of data in Figure 1 can be discriminated by F statistics for main and interaction effects.

Carryover effects make it difficult to specify the sampling distributions of F statistics. Classical F distributions are derived under some assumptions, and while F tests are robust against departures from most of these, learning curves violate an important one: homogeneity of covariance. To see what this means, note that we could calculate a correlation between the four data points in the A_1, t_1 cell of Figure 2 and the four in the A_1, t_2 cell. Under homogeneity of covariance, this correlation would be constant for any pair of cells A_k, t_i and A_k, t_j. However, the correlation between performance after t and $t + 1$ training instances is apt to be higher than the correlation between performance after t and $t + 100$ instances, so homogeneity of covariance is apt to be violated. The consequence is that the Type I error probabilities no longer correspond to the given α level (Cohen 1995 (p. 306), Keppel 1973, O'Brien and Kaiser 1985).

So F statistics can represent the effects in Figure 1, nicely, but carryover effects bias the p values of the statistics. Can we salvage ANOVA and F tests? One common tactic is to correct statistics to compensate for biases. The following experiment (and those in Sec. 5) shows that this tactic will not work. We generated learning curves from three different datasets (Chess, RL, and Tic-Tac-Toe; see the Appendix). The results (Figure 3) demonstrate a dramatic increase in Type I error in the case of Algorithm effects, and a decrease for Interaction effects. The histograms demonstrate that the frequencies of these errors depend on the dataset, which implies that one cannot correct the F statistics with a simple adjustment. In particular, the Chess and Tic-Tac-Toe learning curves were generated according the same procedure, their degrees of freedom are identical, and yet their mean rejection rates differ dramatically.

Another way to salvage ANOVA is to somehow find the appropriate sampling distributions for F statistics when homogeneity of covariance is violated. This would allow us to control Type I errors precisely. Our method, discussed in Section 4, yields these sampling distributions, and accurate p values, whether or not homogeneity of covariance is violated. The procedure is based on *randomization* (see, e.g., Cohen 1995, ch. 5). Consider first the null hypothesis that *Algorithm* has no effect on performance. If it were true, then the lines associated with algorithm A_1 in Figure 2 might equally well be associated with A_2, or with any other algorithm. Thus, if we randomly redistribute lines among algorithms, and then calculate F_{alg} in the usual way, we will derive one value of F_{alg} under the null hypothesis that *Algorithm* is independent of performance. For clarity, denote this statistic F^*_{alg} to remind us that it was derived by randomization, that is, shuffling lines, and to distinguish it from the sample statistic F_{alg} that was calculated from the original (unshuffled) data table. If we shuffle the lines again, we will get another, somewhat different value of F^*_{alg}, and if we shuffle 1000 times we can get a distribution of 1000 values of this statistic.

By shuffling lines instead of, say, individual data points among algorithms, we preserve the dependencies among the data points on each line. Said differently, we treat a line as a unit for the purpose of estimating the distribution of F^*_{alg}, so the degree of dependence among the data on a line is irrelevant. As mentioned above, when homogeneity of covariance is violated, comparing F_{alg} to a conventional F

Initialize $c = 0$. Then do 1000 times:

1. Generate a set L of learning curves using C4.5.
2. Partition L randomly into L_1 and L_2 representing two different imaginary algorithms, with $|L_1| = |L_2| = \frac{|L|}{2}$.
3. Perform conventional ANOVA on these data, obtaining the probability p that it is incorrect to reject the null hypothesis that there is no effect of Algorithm on performance.
4. If $p < 0.05$ then increment c.

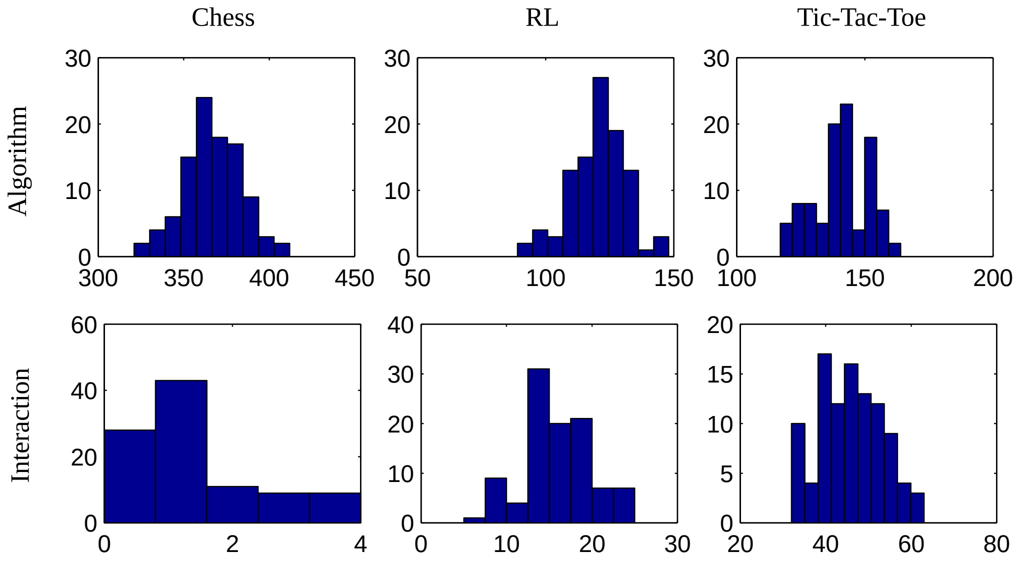

Figure 3: Illustration of the increase in Type I error resulting from carryover effects. For each dataset, the procedure given above was executed 100 times and the resulting c values averaged. Without carryover effects, one would expect $c = 1000\alpha = 50$. The histograms of c values show that H_0 was rejected much more frequently, which demonstrates the inappropriateness of the conventional ANOVA for comparison of learning curves. See the Appendix for details about the datasets used.

distribution will underestimate p, that is, it will make F_{alg} look significant at a given level of α when it is not. The distribution of F^*_{alg} protects against this error, as illustrated by Figure 4.

F^*_{alg} is not technically a sampling distribution but it serves the same purpose, namely, to estimate a p value for a sample result, or to find a critical value that F_{alg} must exceed to reject H_0 with some level α of confidence (Cohen 1995, p. 175).

4 THE PROCEDURE IN DETAIL

Consider a set A of m learning algorithms $A_1, \ldots, A_m$. For each algorithm A_i we have a set $L^{(i)}$ of l learning curves $L_1^{(i)}, \ldots, L_l^{(i)}$. Each learning curve $L_j^{(i)}$ constitutes a k-tuple $(L_{j,1}^{(i)}, \ldots, L_{j,k}^{(i)})$ of real numbers, where each $L_{j,h}^{(i)}$ gives the performance score of the learning algorithm A_i on the jth run after A_i has performed an amount t_h of training.[1] Note that k and the t_h $(1 \le h \le k)$ are the same for all algorithms, but l, the number of learning curves generated by an algorithm, need not be the same for all algorithms.

We will test two null hypotheses: There is no effect of *Algorithm* on performance, and there is no effect of *Algorithm* on the relationship between *Training* and performance. These correspond to F tests of a main effect and the interaction effect in a two-way analysis of variance, so we will compute the appropriate statistics, F_{alg} and F_{int}, but we will compare them to the randomized sampling distributions of F^*_{alg} and F^*_{int}.

The complete procedure can be summarized as follows:

1. For each algorithm i, collect l learning curves $L_1^{(i)}, \ldots, L_l^{(i)}$. If there are m algorithms, this will pro-

[1]The "amount of training" is an abstract notion here which could be given by the number of training instances processed, the number of trials run, or even by the training time.

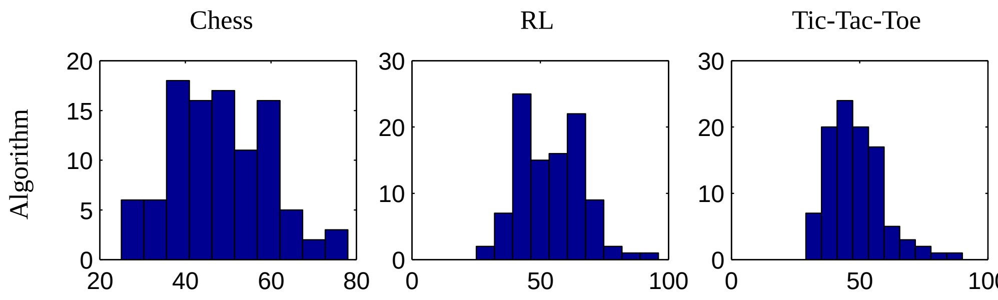

Figure 4: Histograms generated by the same procedure as Figure 3, but p-values were compared against randomized F distributions (500 shuffles) instead of the parametric distributions. In fact, the mean rejection rates of around 50 correspond to the target significance level of $\alpha = 0.05$. This is also true for the corresponding histograms for the Interaction effect (not shown).

duce a data table like the one in Figure 2.

2. Run a conventional two-way analysis of variance on this data table to obtain sample statistics F_{alg} and F_{int}.

3. Generate the sampling distributions F^*_{alg} and F^*_{int}:

   Throw the $m \times l$ learning curves into a "pool" $\mathcal{P}$.

   Do $i = 1 \ldots z$ times (where z is large, e.g., 1000):

   (a) Shuffle $\mathcal{P}$ and reassign each of the ml learning curves to the m algorithm categories (rows in the data table) such that each row contains l curves. Shuffling $\mathcal{P}$ enforces the null hypothesis of no association between performance and algorithm.

   (b) Run a conventional two-way analysis of variance on the resulting data table and record $F^*_{\text{alg},i}$ and $F^*_{\text{int},i}$.

4. Find the critical values in the distributions F^*_{alg} and F^*_{int}. If $\alpha = .05$ and $z = 1000$ then the critical value in each *sorted* distribution is the 950th, because 5% of the distribution lies above this value. In general, the critical value is the $\alpha 100$th quantile.

5. If F_{alg} exceeds the critical value for the F^*_{alg} distribution, reject the null hypothesis that *Algorithm* does not affect performance. Similarly if F_{int} exceeds the critical value for the F^*_{int} distribution, reject the null hypothesis of no interaction effect.

6. The p value for each hypothesis is derived from the rank of the closest value in the sorted sampling distribution. For example, if $F_{\text{alg}} = 10.3$ and the closest value in F^*_{alg} is 10.2, and if the rank of this value is 972 out of 1000, then $p < (1000 - 972)/1000 = .028$.

5 EXPERIMENTAL RESULTS

In Section 3 we illustrated the increase in Type I error caused by comparing F statistics to standard F distributions. This section provides a more detailed account of this phenomenon. Both Algorithm and Interaction effects are analyzed on the Chess dataset (see Appendix). The following section discusses the probability of Type I error, and Section 5.2 compares the power of the conventional and randomized ANOVAs. In all cases we use $m = 2$ sets of learning curves. Note that our method applies to any $m \geq 2$.

5.1 TYPE I ERROR MEASUREMENTS

As shown in Section 3, the standard F distributions tend to overestimate the significance of Algorithm effects, but underestimate the Interaction effects. We expected the overestimations based on previously published results (e.g., Keppel 1973, p. 464) but the underestimations were a surprise and we do not have a satisfactory explanation for this phenomenon. In one sense, we do not care why the standard F distributions detect Interaction effects less often than expected, because we have a method to construct correct F distributions. Yet we were curious. To shed some light on this issue, we examined the frequency of Type I errors for Interaction and Algorithm effects, for conventional ANOVA and our method, in a variety of conditions.

Recall that Type I error rates are the frequencies with which the null hypothesis is rejected when it is true, i.e., when there is no effect. In Section 3 we enforced the null hypothesis by splitting a set of learning curves generated by *one* algorithm into two groups, calling one group "algorithm A," the other "algorithm B," then testing for an Algorithm effect and an Interaction effect. Because the two groups were generated by one algorithm, we expected neither effect; that is, we expected Type I error rates of α. In the following experiments we enforce the null hypothesis in a slightly different way. First we generated a set L of learning curves with C4.5, then to each curve we applied a transformation, yielding another set L'. The transformation induced an Algorithm effect or an Interaction effect or both. In other words, the mean curves for L and L' correspond to the pairs of curves in Figure 1. Then, to enforce the null hypothesis, we shuffled the curves in L and L'. Whereas the earlier procedure enforced the null hypothesis by randomly dividing a set of statistically-identical learning curves, this procedure is more natural in starting with two sets of curves (L and L') that *are* different, then shuffling them. Moreover, we have tight control over the degree of difference between L and L' because we transform the former to get the latter.

We now describe this procedure in detail. The following steps compute the number c of rejections of H_0 during 1000 analyses of variance, starting from a set L of learning curves:

Initialize $c_{\mathrm{conv}} = c_{\mathrm{rand}} = 0$. Then do 1000 times:

1. Construct L' by modifying each curve from L according to one of the cases given in Figure 1. The degree of modification is controlled by a factor f. We will denote this operation by $L' = M_a(L, f)$ for case a in Figure 1, and likewise for cases b, c, d.
2. Partition $L \cup L'$ randomly into L_1 and L_2, with $|L_1| = |L_2| = 20$.
3. Perform conventional ANOVA on these data to obtain the F statistic for the tested effect.
4. Compare F to the appropriate conventional F distribution and read off the probability p_{conv} that it is incorrect to reject H_0.
5. Generate a randomized sampling distribution F^* using 400 shuffles as described in Section 4 item 3, and read off p_{rand}.
6. If $p_{\mathrm{conv}} < \alpha$ then increment c_{conv}.
   If $p_{\mathrm{rand}} < \alpha$ then increment c_{rand}.

This procedure was performed with respect to Algorithm and Interaction effects, and for 10 different values of f. For each of these cases, the c values resulting from 10 such runs were averaged to yield a data point shown in Figure 5. The effect of the modification factor f on the shape of a curve is also illustrated in the figure. Details on the four modification procedures are given in the Appendix.

As expected, the randomized ANOVA always achieves Type I error probabilities near the target significance level of $\alpha = 0.05$. The conventional method, however, tends to assert an Algorithm effect too often (increase in Type I error probability). In contrast, Interaction effects are mostly detected less often than the expected 5%.

Modification M_b is a dramatic case: This modification did not introduce an Algorithm effect, and yet such an effect was often detected by the conventional ANOVA at a frequency inversely proportional to the modification factor f. The modification introduced an Interaction effect which was then shuffled away, enforcing the null hypothesis of no interaction, yet the frequency with which conventional ANOVA detected Interaction effects increases with f. We do not know why, and these experiments fail to explain why Type I errors for interaction effects are lower than expected, although the dependence on f is intriguing.

The magnitude of these misjudgments can be quite dramatic (up to a factor of ten in these examples), but depends on the type of the effect and the modification factor f. Because of these dependencies, we think it is not possible to correct the standard F statistics to control Type I errors precisely. No matter: Our randomized ANOVA produces the expected Type I errors.

5.2 POWER MEASUREMENTS

Whereas Type I errors involve detecting effects that don't exist, Type II errors involve failing to detect errors that do exist. The *power* of a test is one minus the Type II error rate, that is, the probability of detecting a true effect. To measure the power of both conventional and randomized versions of ANOVA, we employed the same modification strategy as in the previous section. Here, however, L and L' are not shuffled. In other words, L and L' give us controlled Algorithm and Interaction effects. The following procedure measures the power of both ANOVAs to detect these effects:

1. Construct $L_2 = M_x(L_1, f)$, where x is one of $a, \ldots, d$.
2. Generate a randomized sampling distribution F^*, as described in Section 4 item 3, using 500 shuffles of 2×10 learning curves each.
3. $c_{\mathrm{conv}} = c_{\mathrm{rand}} = 0$.
4. Do 100 times:
   (a) Randomly draw a set L'_1 of 10 unique curves from L_1.
       Randomly draw a set L'_2 of 10 unique curves from L_2.
   (b) Perform conventional ANOVA and obtain F.

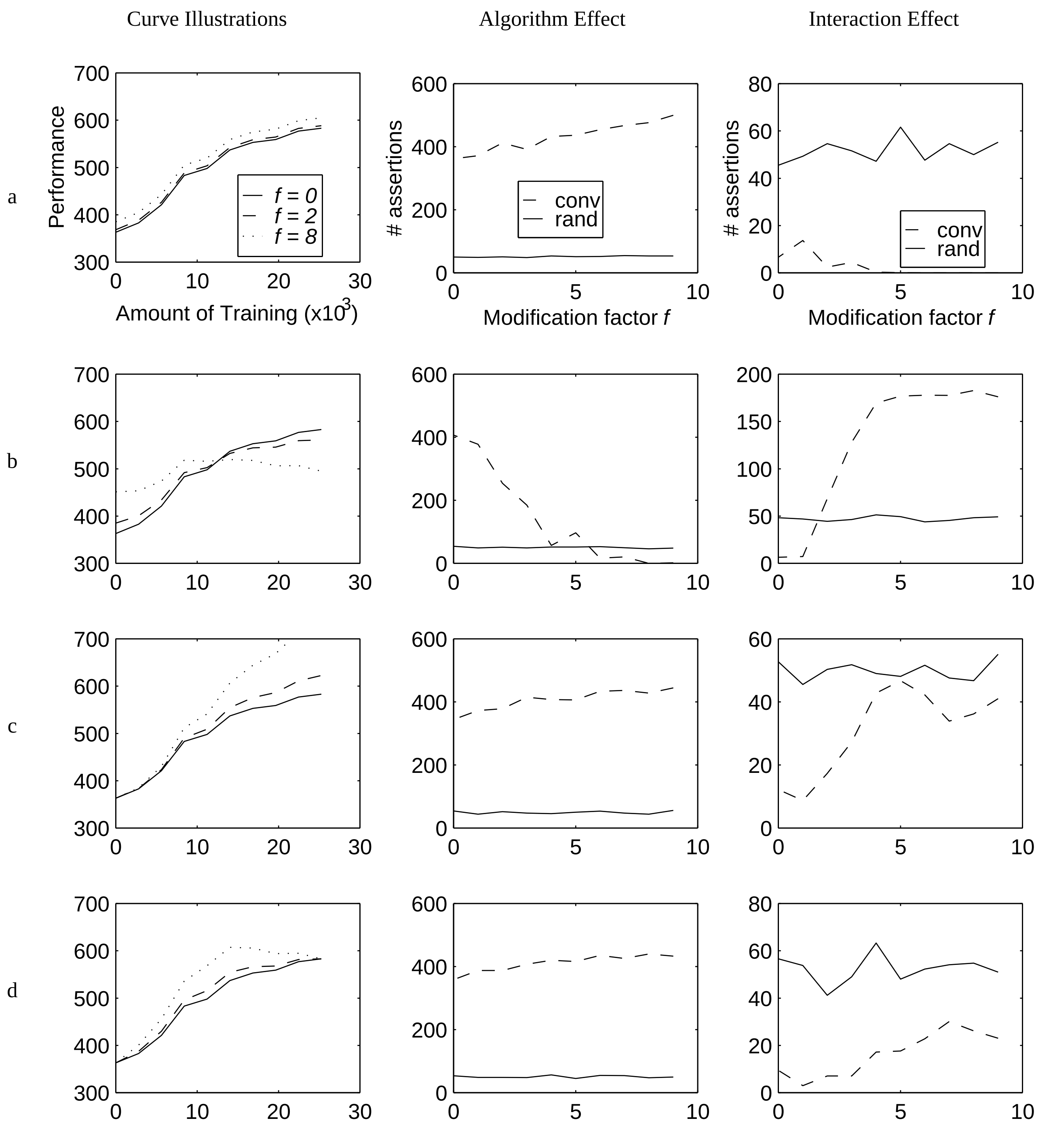

Figure 5: Effects asserted by the conventional and randomized ANOVA methods. Each row shows one of the modification cases a–d from Figure 1. The left column illustrates the effect of the modification for different values of f ($f = 0$ means no modification). The center and right columns plot the number of times (of 1000) the conventional and randomized analyses asserted an Algorithm or Interaction effect at $\alpha = 0.05$.

(c) Compare F to the parametric F distribution and obtain p_{conv}.
Compare F to the randomized F^* distribution and obtain p_{rand}.

(d) If $p_{\mathrm{conv}} < \alpha$ then increment c_{conv}.
If $p_{\mathrm{rand}} < \alpha$ then increment c_{rand}.

Divide c_{conv} and c_{rand} by 100 to obtain the power measurements.

This procedure was performed to introduce Algorithm and Interaction effects for 10 different values of f. For each of these cases, the c values resulting from 8 such runs were averaged to yield a data point shown in Figure 6.

As in earlier experiments, the conventional ANOVA usually overestimates the presence of an Algorithm effect, thus it appears more powerful than our randomized ANOVA. But this "power" is illusory, like a watchdog that barks all night whether or not a prowler is on the premises. Sure, the dog will bark when there is a prowler — the probability of detecting a prowler is 1.0 — but it is a useless animal. In modifications a, c and d, where Algorithm effects are present, our method detects them handily and at a Type I error rate of approximately 5%. In case b, where there is no algorithm effect, our method does not report one, but the conventional method does. Similarly, for interaction effects, our method does not detect one in case a, because none exists, and it is quite powerful in the other cases, where interaction effects are present.

6 CONCLUSION

We have presented a statistical method for comparing sets of performance curves, such as learning curves, when points on the curves are not independent, that is, when there are carryover effects and homogeneity of covariance is violated. We demonstrated that in these conditions conventional analysis of variance produces a sometimes dramatic surplus of Type I errors for main (algorithm) effects and a shortfall of Type I errors for interaction effects. Because the magnitude of these surpluses and shortfalls depends on the original dataset, among other things, we do not think they can be corrected by adjusting conventional F statistics. Instead we show how to construct sampling distributions for the F statistics that correct for violations of homogeneity of covariance. With this method, one can control error rates precisely. We recommend the method for its simplicity and hope it will be a helpful addition to the statistical toolbox of the machine learning community.

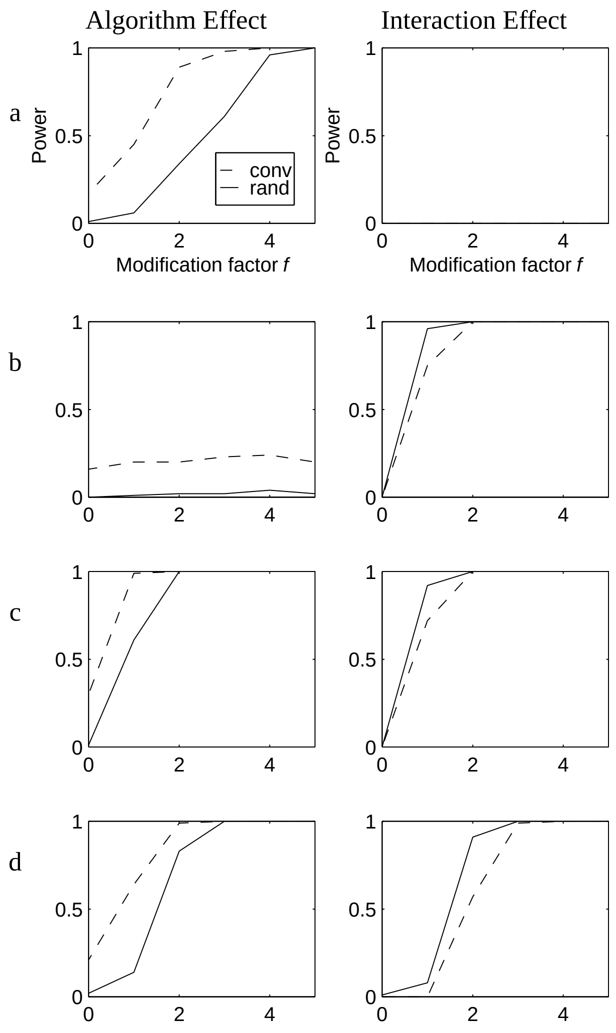

Figure 6: Power measurements of the conventional and randomized ANOVA methods. Each row shows one of the modification cases a–d from Figure 1. The horizontal axes indicate the degree f to which one of one underlying two sets of curves was modified with respect to the other (see Figure 5).

Appendix: Sources of Learning Curves

Chess: Chess Endgame Database (king-rook-vs-king, Bain 1994) provided by the UCI Machine Learning Repository (Merz and Murphy 1996). Twenty Learning curves were generated by running the decision tree algorithm C4.5 (Quinlan 1993) in a 20-fold cross validation procedure.

We now describe the modification functions $M_x(L, f)$ used in Section 5. In the following, r refers to the difference between the performance values of the last and first points of a given learning curve, i.e. $r = L_k - L_1$. For each learning curve L, each performance value L_i is altered according to a given modification case (cf. Figure 1):

(a) $L_i = L_i + f\frac{r}{80}$

(b) $L_i = \begin{cases} L_i + f\frac{r}{100}(\frac{k}{2} - i + 1) & \text{if } i \leq \frac{k}{2} \\ L_i - f\frac{r}{100}(i - \frac{k}{2}) & \text{if } i > \frac{k}{2} \end{cases}$

(c) $L_i = L_i + f\frac{L_i - L_1}{100}(i - 1)$

(d) $L_i = L_i + \begin{cases} fr\frac{i-1}{100} & \text{if } i \leq \frac{k}{2} \\ fr\frac{k-i}{100} & \text{if } i > \frac{k}{2} \end{cases}$

RL: These data were generated by an AI program that employed TD(0) Reinforcement Learning (Sutton 1988) to learn to play Tic-Tac-Toe against a random opponent. The performance score was the cumulative score of one hundred test games against a random player, where losses, draws and wins scored -1, 0, and 1 respectively. Ten learning curves were generated by one training session each.

Tic-Tac-Toe: Tic-Tac-Toe Endgame Database (Aha 1991) provided by the UCI Machine Learning Repository. Learning curves were generated as with the Chess dataset.

References

Aha, D. W. (1991). Incremental constructive induction: An instance-based approach. In *Proc. 8th Int. Workshop on Machine Learning*, Evanston, IL, pp. 117–121. Morgan Kaufmann.

Bain, M. (1994). *Learning Logical Exceptions in Chess*. Ph. D. thesis, University of Strathclyde.

Cohen, P. R. (1995). *Empirical Methods for Artificial Intelligence*. Cambridge, Massachusetts: MIT Press.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation 10*(7), 1895–1924.

Keppel, G. (1973). *Design and Analysis: A Researcher's Handbook*. Englewood Cliffs: Prentice-Hall.

Merz, C. and P. Murphy (1996). UCI Repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

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

O'Brien, R. G. and M. K. Kaiser (1985). MANOVA method for analyzing repeated measures designs: An extensive primer. *Psychological Bulletin 97*(2), 316–333.

Quinlan, J. R. (1993). *Programs for machine learning*. Morgan Kaufmann.

Rasmussen, C. E., R. M. Neal, G. Hinton, D. van Camp, M. Revow, Z. Ghahramani, R. Kustra, and R. Tibshirani (1996). *The DELVE Manual*. University of Toronto, Dept. of Computer Science. http://www.cs.utoronto.ca/∼delve.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning 3*, 9–44.