A Randomized ANOVA Procedure for Comparing Performance Curves

Justus H. Piater and Paul R. Cohen

CMPSCI Technical Report 97-41

July 1997

Computer Science Department
Lederle Graduate Research Center
University of Massachusetts
Amherst, MA 01003-4601

{piater|cohen}@cs.umass.edu

A Randomized ANOVA Procedure for Comparing Performance Curves

Justus H. Piater and Paul R. Cohen

Abstract— Three factors enter into 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, whether the effect of training on performance depends on the algorithm, and whether these effects are localized in regions of the curves. Analysis of variance is adapted to answer these questions. The carryover effects of learning violate the assumptions of parametric analysis of variance, but they are rendered harmless by a novel, randomized version of the analysis. After a brief outline of the statistical preliminaries, we present the procedure along with some examples on real learning curves, discuss power and Type I error, and give some examples of how our method can be applied to answer more advanced questions in comparing performance curves.

1 Introduction

Evaluation of machine learning algorithms typically involves learning curves that plot the amount of training versus the performance of the algorithm. A common question is whether the learning curves generated by two (or more) algorithms are different. These differences can be characterized in terms of the following 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?

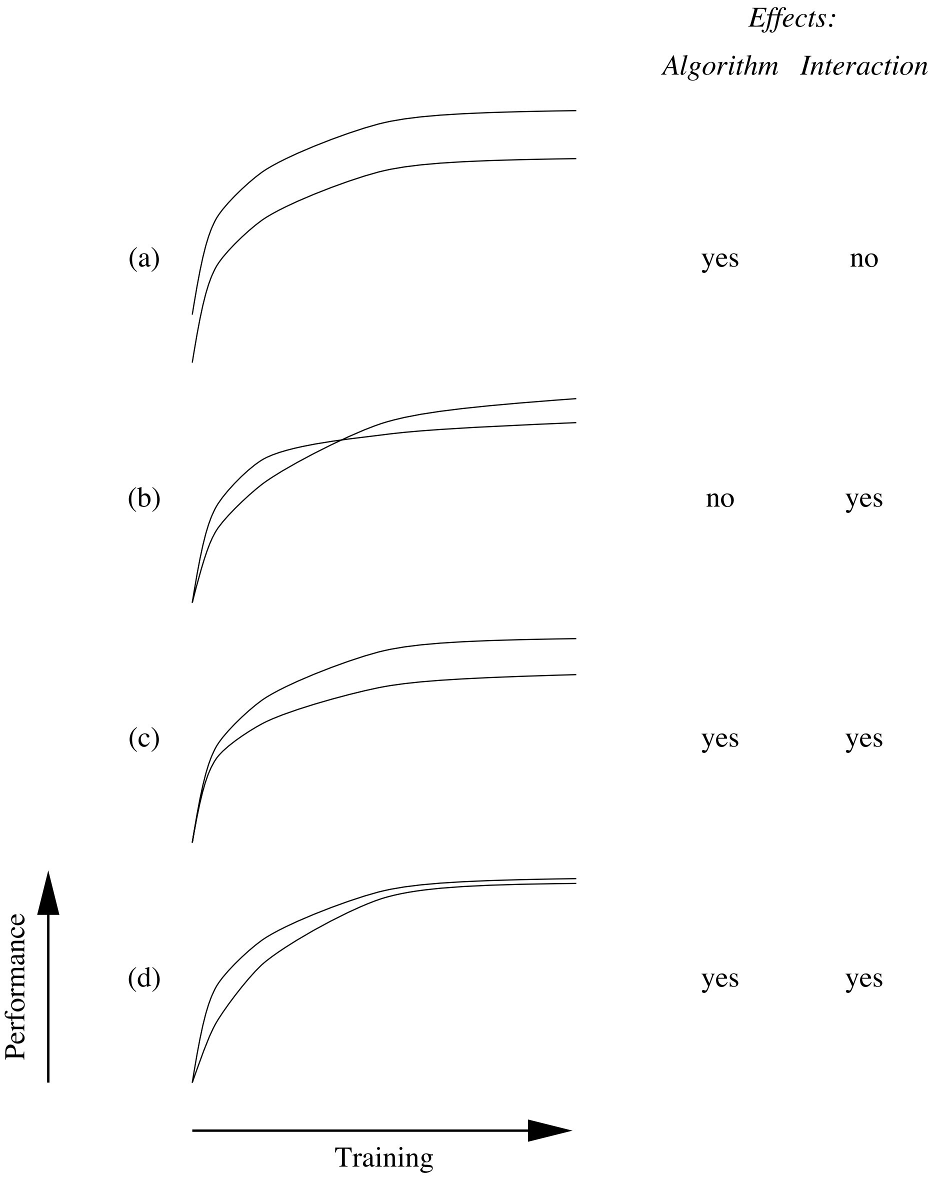


Figure 1: Some kinds of differences between learning curves. The columns on the right indicate the presence of an Algorithm or Interaction effect: (a) Algorithm effect only; (b) Interaction effect only; (c), (d) both effects. In case (c), the Interaction effect disappears at the later stages of training; in case (d), both effects disappear.

Figures 1a and 1b illustrate prototypical cases for each effect. In practice, however, some combination of both effects will occur. In Figure 1c, for instance, both curves start out at 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 after the amount of training exceeds some threshold t_h.

The purpose of this paper is to present a method for detecting and localizing the presence of Algorithm and Interaction effects among curves generated by different algorithms. This method is not restricted to learning curves, but applies to any kind of performance curves. Our methods test two null hypotheses (see the following section for an introduction to hypothesis testing):

- The mean performances of each algorithm A_i are the same (no Algorithm effect).
- The relationship between training t_h and performance does not depend on Algorithm (no Interaction effect).

We also want to answer the following question:

- What fraction of observed Algorithm and Interaction effects can be assigned to a particular training interval?

2 Hypothesis testing and Analysis of Variance

Suppose we have two learning algorithms A_1 and A_2, each of which trains on a set of k instances in a 10-fold cross validation procedure. Then we have ten estimates of the performance of each algorithm at each level of training. Alternatively, we have ten "lines" $L_1^{(1)}, \ldots, L_{10}^{(1)}$ for A_1 and another ten lines $L_1^{(2)}, \ldots, L_{10}^{(2)}$, where each line is a list of k numbers that represent the performance of the algorithm at level $1 \leq h \leq k$ 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 instance, in the experiments reported below, each line is a training session of a reinforcement learning algorithm. The important thing is that the data points on a line are not independent. In statistical parlance, they are "repeated measures" and there is a "carryover effect," meaning that the performance represented by earlier points on a line influences, or carries over, to later performance.

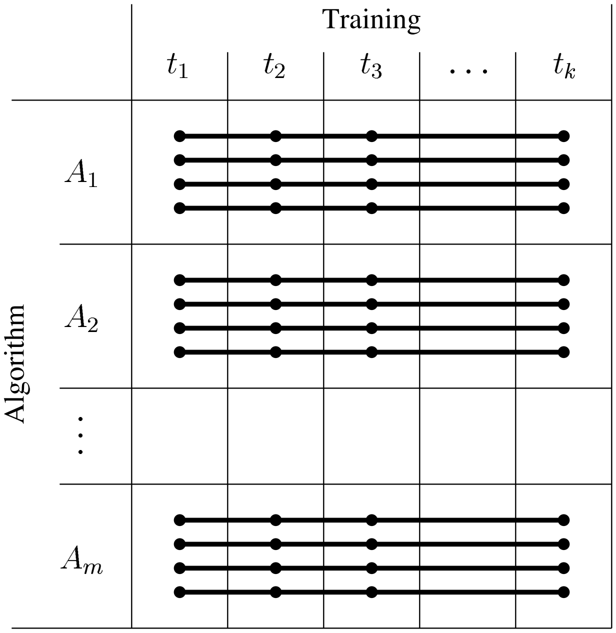


Figure 2: Data table setup for randomized ANOVA. This example shows $l = 4$ learning curves per algorithm.

Were it not for these carryover effects, *analysis of variance* (ANOVA) would be an ideal tool to analyze learning curves. Analysis of variance tests for *main effects* of factors and *interaction effects* between factors. As is common in statistical hypothesis testing, hypotheses are stated in terms of null effects, and the procedure is to reject these null hypotheses if the probabilities of sample statistics under the null hypotheses are very low. The hypotheses, then, are

- There is no effect of training.

- There is no effect of algorithm.
- Any effect of training is constant across algorithms, and any effect of algorithm is constant at all levels of training.

Briefly, the logic of hypothesis testing is analogous to proof by contradiction: To "prove" a proposition P, one shows that the state of the world is inconsistent with not P. In statistics, we substitute "improbable" for "inconsistent": The state of the world, as represented by a statistic's value, is very unlikely to have arisen if not P were true, so not P probably isn't true. More formally, hypthesis 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 S of all possible values of $\phi(i)$ for samples i of size n *under* H_0. These restrictions important: 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. 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 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.

To illustrate, suppose we have a coin c that we suspect is biased, and we want to test this statistically. Let H_0 be that c is fair, and let $\phi(c) = 16$ be the number of times that c lands heads in a sample of $n = 20$ tosses. Let S be the probability distribution of $\phi(i)$ where i is a *fair* coin, thus S is the sampling distribution of $\phi(i)$ under H_0 (it happens to be a binomial distribution with .5 and 20 as parameters). The probability of $k \geq 16$ heads in 20 tosses can be calculated from S; it is approximately $p = .006$. The critical value for this experiment, for $\alpha = .05$ is $\phi(s) = 14$; that is, if c lands heads 14 times or more then one can reject H_0 with a probability of error less than $\alpha = .05$.

It is easy to specify *F statistics* that measure main effects and interaction effects. However, because of the carryover effects it is not so easy to specify the sampling distributions for these 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 effect of violating homogeneity of covariance is to underestimate p values, or, equivalently, to underestimate the probability of asserting an effect when there is no effect. Authors differ on the seriousness of this underestimation (Cohen 1995 (p. 306), Keppel 1973, O'Brien and Kaiser 1985).

The homogeneity of covariance problem can be sidestepped, however, and accurate p values can be obtained, by deriving sampling distributions for F statistics that take the nonindependence of learning curve data into account. The procedure is called 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. It is known that when homogeneity of covariance is violated, comparing F_{alg} to a conventional F 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.

F^*_{alg} is not technically a sampling distribution but it serves some of the same purposes, 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. Conventional sampling distributions test hypotheses about populations (e.g., the hypothesis that in an infinite number of trials, there would be no difference between the mean performance of one algorithm and another). Randomized sampling distributions say nothing about populations, so the null hypothesis is that *Algorithm* is *independent* of performance, on this data set. One should not lose sight of this important difference between classical sampling distributions and randomized sampling distributions (Cohen 1995, p. 175).

3 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, l and the t_h $(1 \leq h \leq k)$ are 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 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 produce 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}.

[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.

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

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

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

   (a) Shuffle 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 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 α100th 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 for F_{int}.

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 $p < (1000 - 972)/1000 = .028$.

Note that for small l and very small m, one can save time (and gain accuracy) by performing exact randomization, i.e. computing the F values of all possible assignments of curves to categories. There are $c_{m,l}$ such assignments, where

$$c_{m,l} = \begin{cases} 1 & \text{if } m = 1 \\ \frac{\binom{ml}{l}}{m} c_{m-1,l} & \text{if } m > 1 \end{cases}$$

However, this number grows very rapidly with m. For example, $c_{2,7}$, $c_{3,4}$ and $c_{4,3}$ are all greater than 1000.

4 Experimental results

Unless otherwise mentioned, all examples in this section use real learning curves obtained from a toy problem (see appendix). The number of curves per algorithm was $l = 10$, and $k = 8$ levels of training were used (0, 200, 500, 1000, 2000, 3000, 5000, 8000). The randomized distributions each consisted of 1000 values.

4.1 ANOVA tables

The first example consists of the two sets of curves shown in Figure 3. The ANOVA table generated by the randomized procedure is shown in Table 1. We find that A_1 and A_2 are different with a high degree of confidence: The algorithm effect is significant at $p = .015$. (The interaction effect is not significant.) The F distributions are shown in Fig. 4. The standard F distributions for the appropriate degrees of freedom (7 and 144 for F_{int}; 2 and 144 for F_{alg}) look essentially the same, but have different means ($\bar{F}_{\text{int}} = 1.02$, $\bar{F}_{\text{alg}} = 1.06$). If compared to the standard F distributions, the ANOVA would return a $p_{\text{int}} = 0.33$ and $p_{\text{alg}} = 0.0016$, the latter of which is a crude overestimation of the significance.

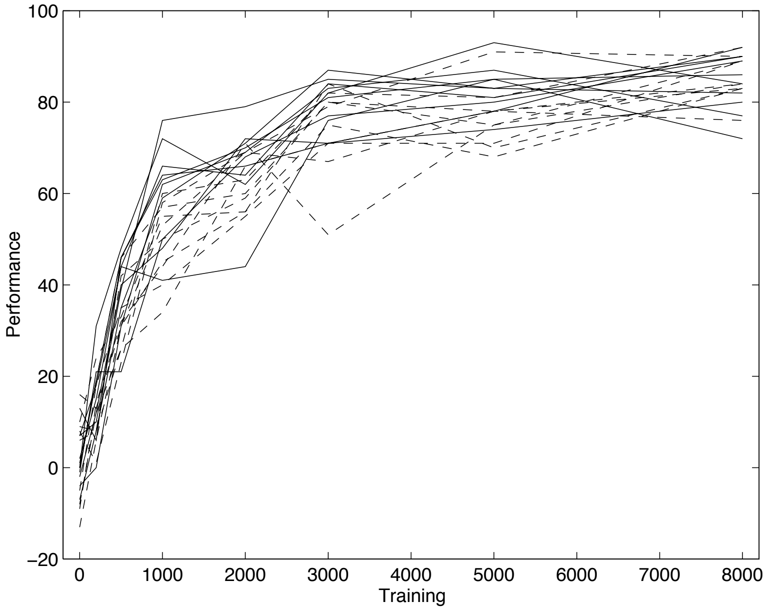
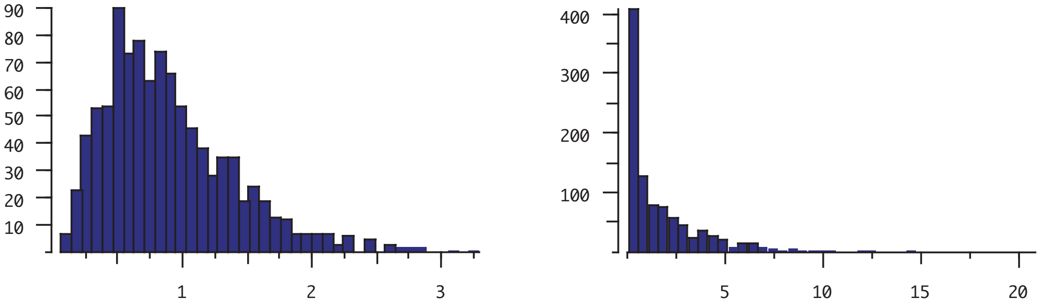


Figure 3: Ten learning curves each from two different example algorithms A_1 (solid lines) and A_2 (dashed lines).

The next example compares algorithm A_1 with A_3 which turn out to be indistinguishable (Fig. 5 and Tab. 2). Like in the previous case, a compari-

	df	SS	MS	F	p
Interaction	7	490.60	70.09	1.16	0.253
Algorithm	1	624.10	624.10	10.31	0.015
Training	7	139019.30	19859.90	328.17	–
error	144	8714.40	60.52		
total	159	148848.40			

Table 1: ANOVA table for learning curves $L^{(1)}$ and $L^{(2)}$ (Fig. 3).



Interaction (mean = 0.905)

Algorithm (mean = 1.837)

Figure 4: Histograms of the F distributions corresponding to Tab. 1.

son of the F statistics to the standard F distributions would underestimate the significance of the interaction effect ($p_{\text{int}} = 0.56$) and overestimate the algorithm effect ($p_{\text{alg}} = 0.645$).

	df	SS	MS	F	p
Interaction	7	5.70	0.81	0.83	0.484
Algorithm	1	0.21	0.21	0.21	0.724
Training	7	1231.47	175.92	179.91	–
error	144	140.81	0.98		
total	159	1378.19			

Table 2: ANOVA table for learning curves $L^{(1)}$ and $L^{(3)}$ (Fig. 5).

A third example compares these three sets of curves in one single analysis. The result is a little less significant than if only $L^{(1)}$ and $L^{(2)}$ are compared, because their difference is shadowed by the close similarity of $L^{(1)}$ and $L^{(3)}$ (Tab. 3).

To illustrate an Interaction effect without the presence of an Algorithm effect,

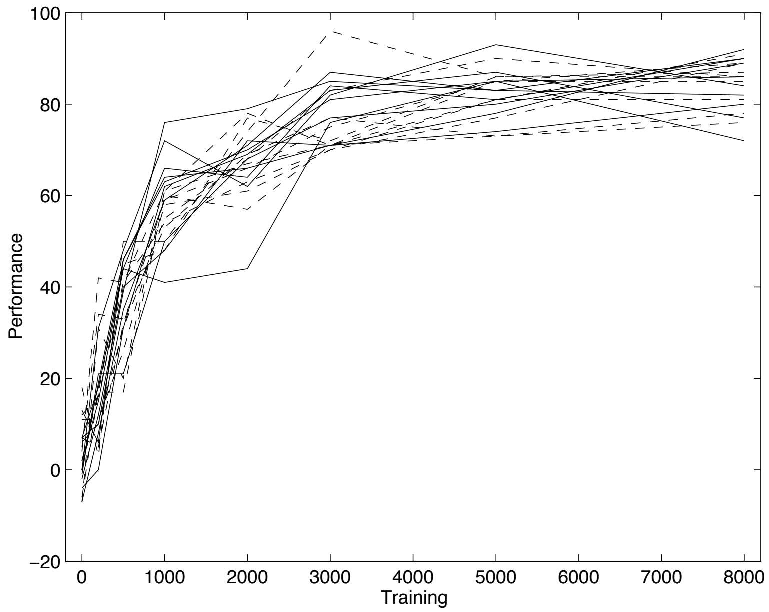


Figure 5: Graphs of learning curves $L^{(1)}$ (solid lines) and $L^{(3)}$ (dashed).

	df	SS	MS	F	p
Interaction	14	771.69	55.12	0.88	0.477
Algorithm	2	799.31	399.65	6.41	0.026
Training	7	203039.10	29005.59	465.58	–
error	216	13456.90	62.30		
total	239	218067.00			

Table 3: ANOVA table for learning curves $L^{(1)}$, $L^{(2)}$ and $L^{(3)}$

we created a dataset $L^{(1a)}$ by adding the row vector $v = [-7, -5, -3, -1, 1, 3, 5, 7]$ to each learning curve of $L^{(1)}$. Because the elements in v sum to zero, datasets $L^{(1)}$ and $L^{(1a)}$ differ only in their column means, but not in their row means. This is the situation shown in Figure 1b. The resulting ANOVA table is given in Table 4. The interaction between algorithm and training is significant at the 0.057 level. This significance increases with the magnitudes of the elements of v. Because of the randomness in the generation of the distribution of F_{alg}, p_{alg} slightly differs from 1.

	df	SS	MS	F	p
Interaction	7	840.00	120.00	1.99	0.057
Algorithm	1	0.00	0.00	0.00	0.984
Training	7	162558.70	23222.67	386.03	–
error	144	8662.80	60.16		
total	159	172061.50			

Table 4: ANOVA table for learning curves $L^{(1)}$ and $L^{(1a)}$

4.2 Performance Measurements

The performance of a statistical test is usually measured in terms of the Type I error and the power (cf. Section 2). To illustrate the power of our method, we computed power curves using 100 learning curves generated by algorithm A_1, here again denoted $L^{(1)}$. A second set $L^{(1')}$ is created by multiplying each item in each learning curve of $L^{(1)}$ by a constant stretch factor s. This yields a useful test situation because the learning curves of both sets start out at roughly the same location on average, but then grow at different rates, which is a situation commonly encountered in Machine Learning.

To generate a point in a power curve for the Algorithm effect, the following procedure was executed:

1. Generate a randomized F^*_{alg} distribution under the null hypothesis by repeating for $i = 1 \ldots 10000$:
   (a) Randomly draw two disjoint, unique samples of appropriate size from $L^{(1)} \cup L^{(1')}$.
   (b) Compute statistic $F^*_{\text{alg},i}$.

2. Obtain the 0.05 critical value c_{alg} from the distribution of F^*_{alg} by averaging 21 values (the 95th percentile and the ten values preceding and following it, to increase accuracy).

3. Initialize r to zero.

4. Do 100 times:

   (a) Draw a set $L^{(a)}$ of l unique curves randomly from $L^{(1)}$.

   (b) Draw a set $L^{(b)}$ of l unique curves randomly from $L^{(1')}$.

   (c) Compute F_{alg} on $L^{(a)}$ and $L^{(b)}$.
   If $F_{\text{alg}} > c_{\text{alg}}$, then increment r by one.

5. The power is given by $r/100$.

Likewise, power curves were generated for the Interaction effect. In Fig. 6, the number of learning curves $l = 10$ was held constant, and the stretch factor s was varied. In Fig. 7, $s = 1.1$ was held constant, and the number of learning curves l per algorithm was varied. Note that the point $s = 1.1$, $l = 10$ occurs in both graphs.

The power increases rapidly with s and l and reaches high values early. For example, in our case, if the performance values of one algorithm exeed those of the other by 10 percent on average, and we have 10 curves in each set, then our procedure will detect an algorithm effect about 80 percent of the time with a Type I error of .05.

5 Further Analysis of Learning Curves

The randomized analysis of variance presented in the previous sections helps us identify effects of *Algorithm* and also interactions between *Algorithm* and *Training* on performance, but it does not tell us which of several algorithms is "best," nor whether two among several algorithms are significantly different, nor whether performance differs in particular regions of two (or more) learning curves. Questions of this sort, which involve comparing the means of two or more cells in the data table in Figure 2, are handled by *pairwise comparisons of means* or, more generally *contrast analysis.* The challenge is to properly estimate the probability of Type I errors, in which the null

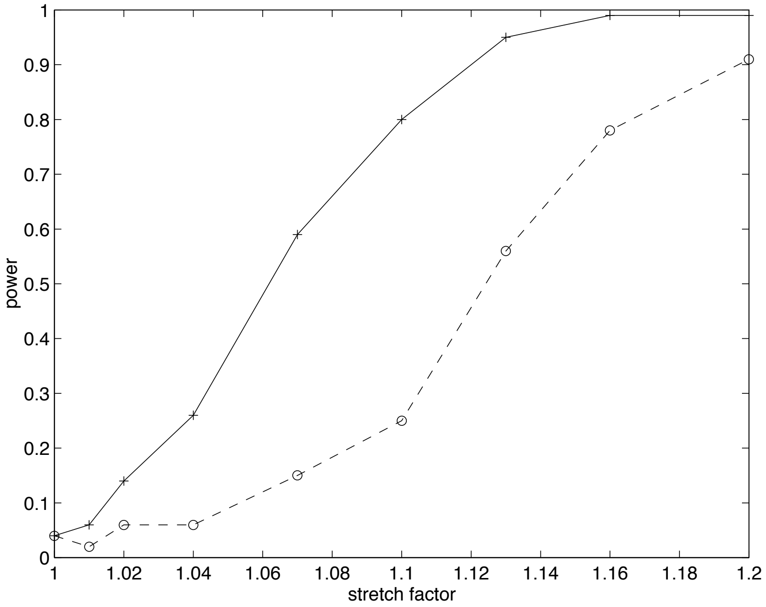


Figure 6: Plot of the stretch factor s versus power of the F_{alg} (solid curve) and F_{int} (dashed) statistics. The number of learning curves per algorithm is $l = 10$, and the Type I error 0.05.

hypothesis is incorrectly rejected and an effect is incorrectly asserted. If one asks many questions of one's data, for instance by comparing the mean performances of every pair of algorithms, and if each comparison has a 0.05 probability of Type I error, then the probability of at least one Type I error in m comparisons is $1 - (1 - 0.05)^m$. Techniques for guarding against the underestimation of total Type I error are discussed in Cohen (1995), chs. 7 and 7A.

It is often the case that learning curves for algorithms A_1 and A_2 are dissimilar at low levels of training but indistinguishable after a lot of training. Alternatively, the curves might rise together, then separate after more training. It would be helpful to know the point at which the difference between A_1 and A_2 goes away. The interesting thing about F statistics is that they are composed of sums; for instance, one term in F_{alg} is the squared effects of algorithm, summed over levels of training. This sum is called SS_{alg} for "sum of squares for algorithm." (You can see the sums of squares in the ANOVA tables presented earlier.) This means we can quit summing after some level of training and ask what proportion of the sum of squares for *all* levels of

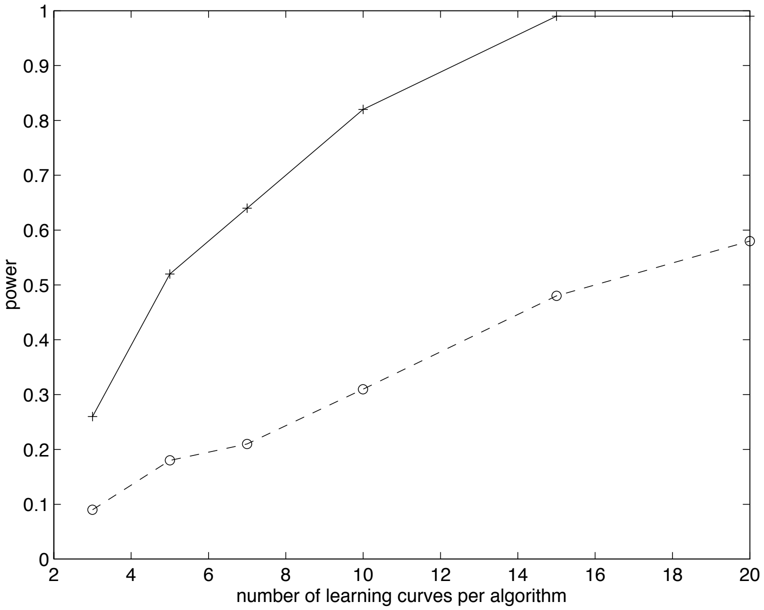


Figure 7: Plot of the number l of learning curves per algorithm versus power of the F_{alg} (solid curve) and F_{int} (dashed) statistics. The stretch factor is $s = 1.1$, and the Type I error 0.05.

training is the sum for just part of the training. If the proportion is very high, then it suggests that the curves differ at low levels of training and not at later levels. Conversely, if the sum of squares for early training is a small fraction of the sum for all training, it suggests that the differences between the curves are most pronounced later in the curves.

The same trick works for interaction effects. If the curves cross early and then run roughly parallel, then SS_{int} summed over the early levels of training will be a large fraction of the total SS_{int}.

6 Conclusion

We have proposed a statistical method for comparing sets of learning curves. Based on a randomized version of two-way analysis of variance, it detects Algorithm and Interaction effects with a given probability of Type I error. Experiments on real data indicated high power to detect existing distinctions. Our method avoids the problem of multiple pairwise comparisons and the homogeneity of covariance problem. We recommend it for its simplicity and hope it will be a helpful addition to the statistical toolbox of the machine learning community.

References

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

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

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.

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

A Learning curves used in Sec. 4

The learning curves used in the above examples were generated by an AI program that learned to play TIC-TAC-TOE against a random opponent. The learning method used was TD(0) Reinforcement Learning (Sutton 1988). The t_h mentioned above refer to the number of training games played. 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. The algorithms differed in their learning rates which were 0.001, 0.8 and 0.1 for A_1, A_2 and A_3 respectively (note the robustness of the learning process with respect to the learning rate).