Decision Criteria for Model Comparison Using the Parametric Bootstrap Cross-Fitting Method

Holger Schultheis

Cognitive Systems Group, University of Bremen, Germany

Ankit Singhaniya

Visvesvaraya National Institute of Technology, Nagpur, India
Oracle India Pvt. Ltd., Bangalore, India

Abstract

When computational cognitive models are compared regarding their ability to fit empirical data, it is important to consider the models' complexity. The parametric bootstrap cross-fitting method (PBCM, Wagenmakers, Ratcliff, Gomez, & Iverson, 2004) is a promising approach to model comparison and selection that takes the compared models' complexity into account. Applying the PBCM requires solving a classification problem, in which it needs to be determined whether a goodness of fit value generated from the compared models is more likely under one or the other of two existing distributions. Previous literature on the PBCM provides little explicit information on (a) the properties of the distributions one should expect to arise in the scope of the PBCM or (b) which methods for solving the classification problem may be suitable (in which situations). This lack of information may hamper use of the PBCM by cognitive modelers. As part of our general endeavor to make sophisticated modeling methods more available and accessible to cognitive scientists developing computational models, in this article we provide detailed analyses of both the distributions that can be expected to arise when employing the PBCM and the performance characteristics of 8 classification methods. Simulation studies involving 6 artificial pairs of distributions and pairs of distributions arising from 8 pairs of existing cognitive models indicate (a) that the relative location but not the shape of the two distributions can be expected to be constrained and (b) that the k-nearest neighbor method constitutes a good general choice for solving the classification problem.

Keywords: Model comparison; Cross-fitting method, Classification, Empirical distributions

1 Introduction

If several computational models are available as potential explanations for the mechanisms underlying certain cognitive phenomena, a common aim is to select that model that is considered to be the best model according to a set of criteria. Besides qualitative criteria such as falsifiability, plausibility, and interpretability (Shiffrin, Lee, Kim, & Wagenmakers, 2008), an important quantitative criterion is how well each model is able to account for empirical data that is pertinent for the phenomena under investigation. A straightforward way of assessing this ability is to fit each model to the relevant data and to measure the achieved goodness of fit (GOF). However, such a straightforward GOF measure may be misleading, because it neglects the complexity of the considered models (e.g., Roberts & Pashler, 2000). As illustrated by Pitt and Myung (2002), for

Corresponding author. Tel.: +49 421 218 64292; Email addressschulth@informatik.uni-bremen.de

example, more complex models may achieve better GOF values than simpler models even if the more complex models do not provide a more accurate account of the mechanisms underlying the phenomena in question. Accordingly, it is important to take model complexity into account when assessing competing models with respect to their GOF and a number of methods are available that are meant to control for model complexity when assessing GOF (see Schultheis, Singhaniya, & Chaplot, 2013; Shiffrin et al., 2008, for overviews).

One of these methods, the *parametric bootstrap cross-fitting method* (PBCM, Wagenmakers et al., 2004), seems particularly interesting for at least two reasons. First, if one of the compared models captures the actual mechanisms that generated the to-be-fitted data, the PBCM has been argued to perform optimally in selecting this model (Cohen, Rotello, & MacMillan, 2008; Shiffrin et al., 2008). Second, the PBCM is applicable to any type of model, since it imposes no constraints on the modeling paradigm or the models' structure. These two properties render the PBCM appealing for model comparison and selection. However, application of the PBCM may be hampered by technical difficulties in employing this method. Although application of the PBCM mainly consists of repeatedly running the competing models, one crucial step requires the solution of a classification problem (see Section 2). Apart from one exception (Cohen, Sanborn, & Shiffrin, 2008) existing literature employing the PBCM (Cohen, Rotello, & MacMillan, 2008; Jang, Wixted, & Huber, 2011; Perea, Gomez, & Fraga, 2010; Wagenmakers et al., 2004) does not provide detailed information on how this classification problem may be solved. Consequently, virtually no information is available on (a) what methods may be used to solve the classification problem and (b) the properties and performance characteristics of possible methods. This lack of information may prevent cognitive modelers from using the PBCM.

As part of our general endeavor to make sophisticated modeling methods more available and accessible to cognitive scientists developing computational models, we provide a systematic investigation of 8 classification methods across a wide range of classification situations. By only considering classification methods that are easy to implement and use even for cognitive scientists without experience in classification, the results of our investigation foster model comparison and selection in at least two ways. First, differential strengths and weaknesses of different methods are identified thus allowing to avoid suboptimal model selection accuracy and unnecessarily long runtimes when employing the PBCM. Second, since explicit information on possible methods is provided, the PBCM becomes more accessible to cognitive modeling researchers. Both aspects, we believe, are helpful for increasing the frequency with which the PBCM instead of GOF assessment not controlling for model complexity will be employed.

We start our considerations with a brief overview of the PBCM and the classification problem involved in its application (Section 2). Subsequently, we describe each of the 8 investigated classification methods and their properties (Section 3). Sections 4 and 5 detail the procedures and results of the simulations we used to assess the methods' performance. In Section 6 we discuss impact of our results on the use of the PBCM, before we close by highlighting main insights and by giving an outlook on future work (Section 7).

2 The PBCM

Let A and B be two competing models and $\mathbf{x}$ a set of observed data (e.g., response times from different experimental conditions). Furthermore, let $\Delta gof_{AB}^{\mathbf{x}}$ be the GOF difference of the two models on the data set $\mathbf{x}$, that is, $\Delta gof_{AB}^{\mathbf{x}} = gof_{A}^{\mathbf{x}} - gof_{B}^{\mathbf{x}}$, where $gof_{A}^{\mathbf{x}}$ and $gof_{B}^{\mathbf{x}}$ are the goodness of fits the models A and B achieve on $\mathbf{x}$, respectively. A naïve approach to assessing how well each model is able to account for empirical data would assume that A provides the better account if $\Delta gof_{AB}^{\mathbf{x}} \geq 0$ and that B provides the better account if $\Delta gof_{AB}^{\mathbf{x}} < 0$. The PBCM aims to improve on the naïve approach by taking into account how well the models are able to mimic each other, that is, the ability of each model to provide good fits to data generated by the other model.

To achieve this, the PBCM generally proceeds as follows:

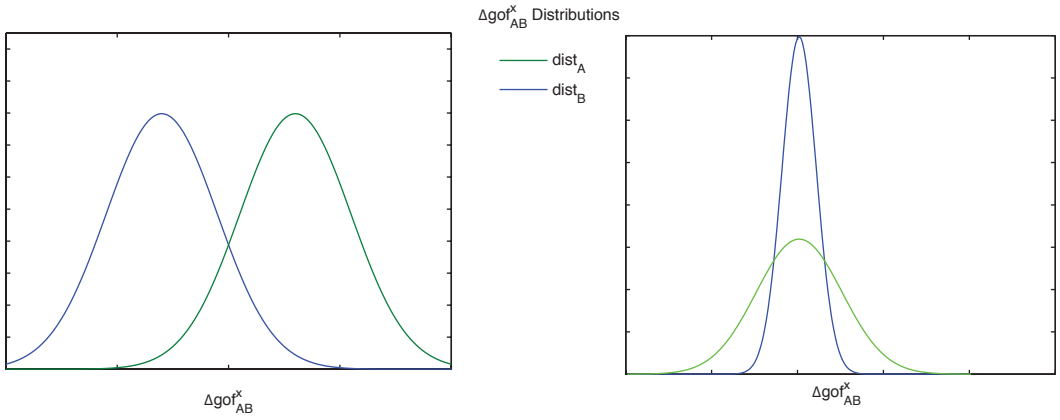


Figure 1: GOF difference distributions as they may arise in using the PBCM. The blue / green curve indicates GOF differences obtained when model A / B have generated data. The left panel shows non-nested distributions, the right panel shows nested distributions with $dist_A$ being nested within $dist_B$.

1. generate a set of parameter values for all parameters of model A,
2. generate a data set $\mathbf{x}_A$ by running model A with the parameter values from the first step,
3. fit both models to $\mathbf{x}_A$ to obtain $\Delta gof_{AB}^{\mathbf{x}_A}$,
4. repeat the above three steps NBS number of times.

These steps will result in NBS many GOF differences for data that has been generated from model A. If the same four steps are repeated with model B as the data-generating model, one obtains a second set of NBS many GOF differences. These two sets of GOF differences constitute two distributions, $dist_A$ and $dist_B$, respectively, that provide information on how well the two models are able to mimic each other (see Figure 1).

In particular, the two distributions can inform model comparison and selection. Distribution $dist_A$ allows to gauge how likely it is to obtain the models' GOF difference on the observed data, $\Delta gof_{AB}^{\mathbf{x}}$, if model A is the generating model. Distribution $dist_B$ allows to gauge how likely it is to obtain $\Delta gof_{AB}^{\mathbf{x}}$, if model B is the generating model. The PBCM selects the model that is associated with the distribution under which $\Delta gof_{AB}^{\mathbf{x}}$ is more likely: If $\Delta gof_{AB}^{\mathbf{x}}$ is more likely under $dist_A$, model A is selected; otherwise, model B is selected. Essentially, this selection procedure amounts to solving a classification problem: Given two classes ($dist_A$ and $dist_B$) and an observation with unknown class membership ($\Delta gof_{AB}^{\mathbf{x}}$), assign the observation to one of the two classes (i.e., select one of the models). Solving this classification problem accurately is critical for the performance of the PBCM.

One question related to the PBCM as described so far is how to sample the parameter values for the data-generating model in step 1 above. Wagenmakers et al. (2004) propose two different ways of generating parameter values. In the first variant, called data-informed PBCM, parameter values are determined based on the data that is to be modeled, $\mathbf{x}$. In the second variant, called data-uninformed PBCM, parameter values are generated independently of $\mathbf{x}$: One first fixes a range of possible values for each model parameter and a probability distribution across each of these ranges. Values for generating data are then sampled from the ranges according to the associated distributions. Since the data-informed PBCM has been argued (Wagenmakers et al., 2004) and shown (Schultheis & Naidu, 2014) to be inferior to the data-uninformed PBCM regarding model comparison and selection, we exclusively employ the data-uniformed variant in our simulations. For the sake of brevity we will refer to the data-uninformed PBCM simply as PBCM in the remainder of this article.

2.1 Expected Distributions

The properties of the two distributions $dist_A$ and $dist_B$ and their relation to each other are of crucial importance both for the optimal classification performance that is achievable and for which methods will be able to perform well. Previous research has not explicitly addressed the question of what kind of distributions may arise in the scope of employing the PBCM. Without specific assumptions on the models or situations that give rise to the distributions, it seems hard (if not impossible) to rule out any type of pairs of distributions: In principle, $dist_A$ and $dist_B$ may be of any distribution type and in any relation to each other.

Nevertheless, we think that there are reasons to expect certain types of pairs of distributions to occur more frequently than others. To see this, consider an idealized situation in which the data generated from each model does not contain any noise and in which the procedure used for fitting the models always returns the best possible fit. In such a situation each model will always fit its own data at least as well as the data generated by the other model:

$$gof_A^{\mathbf{x}_A} \geq gof_A^{\mathbf{x}_B},$$
$$gof_B^{\mathbf{x}_B} \geq gof_B^{\mathbf{x}_A}.$$

These two inequalities together imply that the GOF difference of the two models on data generated by model A will always be at least as large as the GOF difference on data generated by model B:

$$gof_{AB}^{\mathbf{x}_A} \geq gof_{AB}^{\mathbf{x}_B},$$
$$\forall y \text{ such that } \exists gof_{AB}^{\mathbf{x}_B} \geq y \implies gof_{AB}^{\mathbf{x}_A} \geq y, \forall gof_{AB}^{\mathbf{x}_A},$$
$$\forall y \text{ such that } \exists gof_{AB}^{\mathbf{x}_A} \leq y \implies gof_{AB}^{\mathbf{x}_B} \leq y, \forall gof_{AB}^{\mathbf{x}_B}.$$

This means that, in such an idealized situation, the two distributions will either not overlap at all or overlap only in a single GOF difference value.

Due to noise and imperfect fitting, the above inequalities will rarely, if ever, hold in actual modeling situations. Noise, for example, may produce data sets from model A that can be better fitted by model B than by model A. However, as long as noise, imperfect fits, and perhaps further influences do not introduce systematic biases into the obtained GOF differences, the relations formalized by the above inequalities will only be clouded such that they are preserved as tendencies in the obtained distributions. Put formally, unsystematic influences will create distributions for which the following inequalities on probabilities (P) hold:

$$P(gof_A^{\mathbf{x}_A} \geq gof_A^{\mathbf{x}_B}) > P(gof_A^{\mathbf{x}_A} < gof_A^{\mathbf{x}_B}), \tag{1}$$
$$P(gof_B^{\mathbf{x}_B} \geq gof_B^{\mathbf{x}_A}) > P(gof_B^{\mathbf{x}_B} < gof_B^{\mathbf{x}_A}), \tag{2}$$
$$P(gof_{AB}^{\mathbf{x}_A} \geq gof_{AB}^{\mathbf{x}_B}) > P(gof_{AB}^{\mathbf{x}_A} < gof_{AB}^{\mathbf{x}_B}), \tag{3}$$
$$P(gof_{AB}^{\mathbf{x}_A} \geq y) \geq P(gof_{AB}^{\mathbf{x}_B} \geq y), \forall y, \tag{4}$$
$$P(gof_{AB}^{\mathbf{x}_B} \leq y) \geq P(gof_{AB}^{\mathbf{x}_A} \leq y), \forall y. \tag{5}$$

If these relations between probabilities hold, the pair of distributions will be of the kind shown in the left panel of Figure 1: The distributions may overlap, but one of the distributions will (more or less clearly) occupy a region with smaller GOF differences than the other distribution. If noise, imperfect fitting, or other influences introduce one or more systematic biases into the data, however, some or all of the inequalities may be violated such that, in principle, $dist_A$ and $dist_B$ may be of any distribution type and in any relation to each other. For example, one of the distributions may be *nested* within the other in the sense that comparatively small and high GOF difference are more likely under $dist_B$ while comparatively medium GOF differences are more likely under $dist_A$ (see right panel of Figure 1).

Without knowledge to the contrary, it seems more parsimonious to assume that noise, imperfect fitting, and other such factors will commonly not introduce systematic biases into GOF differences. Accordingly, we hypothesized that the distribution pairs arising in the scope of employing the

PBCM will often be of the type for which the inequalities (1) – (5) on probabilities hold.[1] Existing depictions of PBCM distribution pairs seem to support this hypothesis (see, e.g., Figures 5, 7 – 9, and 11 in Wagenmakers et al., 2004) and we explicitly test the hypothesis in more detail on the distributions obtained in our own simulations.

3 Classification Methods

Classification can be a hard task and decades of research have proposed and assessed various methods to solve classification problems (Duda, Hart, & Stork, 2001, provide an overview). In principle, many of the methods considered in classification research such as, for example, artificial neural network classifiers or decision trees could be employed for selecting a model in the PBCM. However, many of the available methods are difficult to apply successfully and, thus, may prevent wider usage of the PBCM in cognitive modeling research.

In line with our aim to facilitate the use of the PBCM, the 8 methods we consider have been chosen to be easy to employ even for non-experts in classification. This choice of methods is also reasonable given that the classification problem that needs to be solved in the PBCM is comparatively simple: Only two classes ($dist_A$ and $dist_B$) with scalar instances (GOF difference values).

The considered methods can roughly be divided into boundary methods (Section 3.1) and direct methods (Section 3.2). To characterize the runtime complexity of the methods we will refer to the number of samples that make up each of the two empirical distributions (NBS) and to the number of samples that need to be classified (NTS).

3.1 Boundary Methods

The two distributions $dist_A$ and $dist_B$ contain information about the properties of the instances belonging to each of the two corresponding classes. Boundary methods attempt to extract this information to establish one or more decision boundaries that yield minimal classification error. If a new instance with unknown class membership has to be classified, the relation of the instance to the decision boundaries is assessed. Based on the result of the assessment the instance is assigned to one of the two classes. When assignment to neither of the two classes is possible, we say that the method *cannot say* (CS) to which class the instance belongs. If, for example, 3.5 was identified as the decision boundary this means that all $\Delta gof^{\mathbf{x}}_{AB}$ that are less than 3.5 will be assumed to belong to one class, all $\Delta gof^{\mathbf{x}}_{AB}$ that are greater than 3.5 will be assumed to belong to the other class, and all $\Delta gof^{\mathbf{x}}_{AB}$ that are equal to 3.5 constitute CS cases.

In this example, only a single boundary has been determined and this seems appropriate whenever one of the distributions occupies a region with smaller GOF differences than the other distribution (see left panel of Figure 1). A single boundary is less appropriate if the two distributions are nested (see right panel of Figure 1). In such a case, two decision boundaries are required to achieve good classification results: The class corresponding to the nested distribution is chosen if $\Delta gof^{\mathbf{x}}_{AB}$ lies between the two decision boundaries, and the method fails to classify if $\Delta gof^{\mathbf{x}}_{AB}$ lies on either of the two decision boundaries.

Flexibility in using either a single or two decision boundaries requires estimating how many boundaries are most appropriate. In our simulations we compare properties of $dist_A$ and $dist_B$ to determine the number of employed boundaries. Unless stated otherwise, if the range from the lowest to the highest GOF difference in $dist_A$ / $dist_B$ completely includes the range from the lowest to the highest GOF difference in $dist_B$ / $dist_A$, we assume that two boundaries need to be determined. In all other cases a single boundary is determined assuming that the distribution containing the lowest GOF difference lies to the left of the other distribution.

We decided to restrict our simulations to distinguishing single and nested, two boundary cases. First, in line with our considerations on the relation of distributions occurring in the scope of the

[1] Note that our considerations and the resulting hypothesis do not say anything about the shape of $dist_A$ or $dist_B$, but only about the relation between the distributions. Even for the idealized situation, any distribution shape seems possible.

PBCM (see Section 2.1), we assumed that situations requiring more than two boundaries will rarely, if ever, arise. Second, it is not immediately clear how a reasonable algorithm for determining more than two boundaries may look like.

The boundary methods we considered are binning, smoothed binning, equidistant search, adaptive search, and Gaussian parametric. Each of these methods is described in the remainder of this section. Further detail on properties and implementation of these methods is provided in Table 1.

3.1.1 Binning

The idea underlying the binning method is to approximate the distributions' densities by accumulating the given samples into k bins and to then determine the decision boundary by finding the intersection(s) of these approximate densities.

The nestedness of the two distributions is determined by examining the binned distributions. If all non-zero bins of one distribution are contained within the range of non-zero bins of the other, the distributions are assumed to be nested. Otherwise, the left and right distributions are identified by the relative occurrence of their peaks.

For non-nested distributions, we determine the boundary as the mid-point of that bin between the left and right peak for which the two binned densities cross over. For nested distributions, we determine the mid-point of two cross over bins: One to the right and one to the left of the peak of the inner distribution. We refer to this method as *Bin*, in this article.

3.1.2 Smoothed Binning

One way to reduce the influence of sampling noise on the approximation of the densities is to smooth the bin values by computing a moving average across N, with N odd, neighbored bins and to replace the sample count of the middle bin with the average.

This method first accumulated the samples into bins and computed a moving average to smooth the bin values. Then it proceeded on the smoothed bins as the Bin method. We refer to this method as *Bin-N*, where N is replaced by the actual number employed for smoothing.

3.1.3 Equidistant Search

One way to search for the optimal boundary is to pick a number of candidate boundaries and evaluate the corresponding classification accuracy for each of these potential boundaries. Equidistant search, which was employed by Cohen, Rotello, and MacMillan (2008), works with K potential boundaries that are equidistantly distributed across the joint range of both distributions.

For non-nested distributions, this method estimated classification accuracy for each of the K candidate boundaries. It chose that point as the decision boundary, which yielded maximum classification accuracy. For nested distributions, the method enumerated all $\binom{K}{2}$ pairs of points (p_1, p_2) such that $p_2 > p_1$. For each of these pairs, the classification accuracy considering p_1 as the left decision boundary and p_2 as the right decision boundary was estimated. The pair of points providing maximum classification accuracy was chosen as the two decision boundaries. This method is termed *E-search-K*, where K is replaced by the number of candidate boundaries employed.

3.1.4 Adaptive Search

This method is an adaptive version of the equidistant search method. It instantiates a local search by iteratively employing equidistant search in regions of major interest. It attempts to decrease the number of potential decision boundaries that need to be considered without sacrificing classification accuracy by focusing search on the most promising regions of the range covered by the distributions.

This method also started out with a set of K equidistant points and the classification accuracy associated with each (pair of) point(s) was determined. The best (pair of) point(s) was used to define

Method	Runtime Complexity	Parameters	Local Search	Technical Complexity
Bin	$O(NBS + NTS)$	$k = NBS/10$	No	Low
Bin-N	$O(NBS + NTS)$	$N = 3, 5, 9$	No	Low
E-search-K	Non-Nested: $O(K * NBS + NTS)$ Nested: $O(K^2 * NBS + NTS)$	$K = 10, 1000$	No	Low
A-search-K	Non-Nested: $O(K * NBS + NTS)$ Nested: $O(K^2 * NBS + NTS)$	$K = 5, 10$ $T = 5 * 10^{-8}$	Yes	Moderate
parametric	$O(NBS + NTS)$	-	No	Low
kernel	$O(NTS * NBS)$	-	No	Low
k-NN	$O(NTS * NBS)$	$k = 10$	Yes	Low
vicinity	$O(NTS * NBS)$	$K = 0.02$	Yes	Low

Table 1: Parameterization and properties of the considered methods

a new range (a subrange of the original range) from which again K equidistant points were chosen. This iterative refinement of the considered range is repeated until an improvement in classification accuracy is achieved or until the size of the considered range drops below a pre-specified threshold T (in our simulation). Of all (pair of) point(s) considered during the search, the one yielding the highest accuracy is chosen to define the decision bound(s).

For non-nested distribution pairs, the subrange are defined as close proximity (1 equidistant point to left and right) to the point that gave maximum classification in the current iteration. In the case of nested distributions, subranges on subsequent iterations are determined by making $p1$ the left range limit and $p2$ the right range limit, where point $(p1, p2)$ gave maximum classification accuracy on the current iteration.

We refer to this method as *A-search-K*.

3.1.5 Gaussian Parametric

This method assumes that the samples come from distributions that can reasonably well be approximated by normal distributions.

Mean and standard deviation of the empirical distribution were derived from the available samples. These empirical means and standard deviations are used to define two normal distributions. The boundaries were then determined analytically from the resulting probability density functions.

Whether the distributions are nested or not directly emerged from computing the decision boundaries. Two normal distribution either have one or two intersection points and the analytical solution generated as many boundaries as there are intersections. We refer to this method as *parametric* in the remainder of this article.

3.2 Direct Methods

Direct methods do not require identification of decision boundaries. Instead, they directly relate each new sample to the samples of the two empirical distributions and classify the new sample based on this relation. As a consequence, (a) there is no need to estimate the number of decision boundaries for direct methods and (b) the information available from $dist_A$ and $dist_B$ needs to be extracted and considered every time a test difference is classified.

The direct methods we considered are the kernel method, the k-nearest neighbor classifier, and the vicinity search. In the following we describe each of these methods. Further detail on properties and implementation of these methods is provided in Table 1.

3.2.1 Using Kernels[2]

This method makes use of Gaussian kernels to determine whether a new sample should be assumed to belong to either of the two distributions.

For each of the two distributions A and B we defined a Gaussian kernel g_a and g_b, respectively, with mean zero and a standard deviation equal to the standard deviation of the corresponding empirical distribution. For each new sample x, we applied the Gaussian kernels as follows:

$$f_a(x) = \sum_{\mathbf{x_i} \in A} g_a(\mathbf{x} - \mathbf{x_i});\ f_b(x) = \sum_{\mathbf{x_i} \in B} g_b(\mathbf{x} - \mathbf{x_i})$$

if $f_a > f_b$, we assumed that the new sample belongs to distribution A, and if $f_a < f_b$, we assumed that it belongs to distribution B. Whenever f_a was equal to f_b, the method failed to classify the sample (i.e., encountered a CS case). This method is called *kernel* henceforth.

3.2.2 k-Nearest Neighbor

This method is a variant of the k-nearest neighbor algorithm (see, e.g., Duda et al., 2001, Chapter 4.5). The idea underlying this method is that a sample likely belongs to that class whose instances are closest to the sample.

To classify a sample x, the k nearest samples from each distribution, $dist_A$ and $dist_B$, are identified and the absolute distance of these samples from x is summed to give a combined distance d_a and d_b, respectively. If $d_a < d_b$, we assume that x belongs to $dist_A$; if $d_a > d_b$, we assume that x belongs to $dist_B$. The method cannot make a decision, when the combined distances d_a and d_b are equal. We refer to this method as *k-NN* in the remainder of this article.

3.2.3 Vicinity Search

This method compares the frequencies of empirical samples from each distributions in a (close) neighborhood of the test sample.

Vicinity search defined a vicinity V of sample x as

$$V = x \pm K * R,$$

where R was the combined spread of two distributions and K was a parameter of the method. The sample x was assumed to belong to that distribution that yielded the higher frequency in V. In case the frequencies were equal, the method was unable to determine to which class x belonged. We term this method *vicinity* in the following.

3.3 Observations on Method Properties

The differences in the methods' approaches to classification imply a number of differences in the methods' expected behavior. In the following we highlight these differences by discussing important properties of each method.

Binning and smoothed binning are conceptually straightforward and appealing, but it can be difficult to find a good balance between the number of bins and bin size: few large bins provide a reliable but coarse approximation while many small bins provide a more detailed but perhaps unreliable approximation. Smoothing may increase reliability, but potentially comes at the cost of smoothing out not only noise but also variations of interest.

The E-Search allows for performance close to the optimum when K is sufficiently high. At the same time, runtime of the E-search may increase quadratically in the value of K. Usually, it will be unclear for a given classification situation which K best balances runtime and classification performance.

[2]We thank Arthur Breitman for suggesting this method.

In comparison to the E-search, A-Search attempts to reduce the number of potential boundaries without compromising on the search precision and thus improves time requirement considerably. Given the adaptive nature of the A-Search, its search is local and, thus, susceptible to getting stuck in local optima. Larger values of K reduce the chance of getting stuck in local optima, but, as for the E-search, what constitutes a good choice for K may be hard to determine.

The parametric method and the kernel method both are expected to achieve satisfactory performance only, if $dist_A$ and $dist_B$ can be approximated by normal distributions. The runtime complexity of the parametric method is considerably lower than that of the kernel method.

The k-NN method attempts a local density estimation of both distributions around the test sample. This can lead to misclassifications whenever the local estimate is unreliable (e.g., due to sampling noise). Such problems may be addressed by varying the value of k, but usually the best choice for the value of k is unknown. As with all direct methods, a potential drawback of the k-NN is that its runtime scales with the number of samples that need to be classified.

Vicinity search combines ideas of the k-NN method (local estimation) with ideas of the binning methods (using a bin for estimation). As a result, it inherits some of the problems associated with these approaches: (a) unreliable local estimates and (b) the difficulty to determine a good bin size.

To what extent and how these properties of the methods will impact their performance when utilized in the scope of the PBCM is a largely empirical question. In the following sections we describe several sets of simulations that address this question.

4 Artificial Distributions

The properties of the two distributions $dist_A$ and $dist_B$ that are associated with the two models A and B depend on the nature of the models A and B and on the situations being modeled, among others. Since it is not clear how the models' and situations' properties map onto distribution properties, it is difficult to control distributional properties when assessing the methods' performance by employing existing cognitive models. Therefore, one part of method assessment considered pairs of distributions with known properties. Instead of generating distributions by repeatedly running and fitting models, we generated distributions by sampling from a number of known distributions. This allowed directly investigating each methods' susceptibility to the distributions' shape, nestedness, and relative spread.

The 6 artificial distribution pairs, S1 – S6, we employed are illustrated in Figure 2. S1 consists of two normal distributions, one with $\mu = -5000, \sigma = 3 * 10^6$ and the other with $\mu = 1000, \sigma = 3$. S2 also comprises two normal distributions, but this time the two distributions have identical spread: $\mu = 0, \sigma = 2$ for the first distribution, $\mu = 2, \sigma = 2$ for the second distribution. One normal ($\mu = 100, \sigma = 10$) and one uniform (over the interval $[-100, 1000]$) distribution are combined in pair S3. Pair S4 also has one normal ($\mu = -70000, \sigma = 100000$) and one uniform (over the interval $[-100000, 0]$) distribution. Pair S5 comprises a log-normal distribution ($\mu = 0, \sigma = 1$) and another log-normal of the same shape, but shifted two units along the x-axis. Pair S6 combines a normal distribution ($\mu = 2.5, \sigma = 4$) with a log-normal distribution ($\mu = 0, \sigma = 1.25$). As a result, pairs S3, S4, and S6 constitute situations with different distribution shapes, pairs S1, S3, and S4 constitute situations with nested distributions, and pairs S1 and S3 constitute situations with extreme differences in relative spread. Pair S6 constitutes a situation in which 3 decision boundaries would be optimal and, more generally, provides a test on the difficulties that may arise due to (incorrectly) estimating the number of required boundaries.

For each of these distribution pairs, method assessment proceeded as follows: First, NBS many samples were drawn from both distributions in the pair. The two resulting sets of samples were treated as analog to $dist_A$ and $dist_B$ in the PBCM (see Section 2). Second, NTS many additional samples were drawn from both distributions in the pair. Based on the two sampled distributions, each method was used to classify the two sets of NTS many samples.

To obtain a more comprehensive view on the methods' performance, it seemed important to assess the methods across different values of both NBS and NTS. In our study, we combined

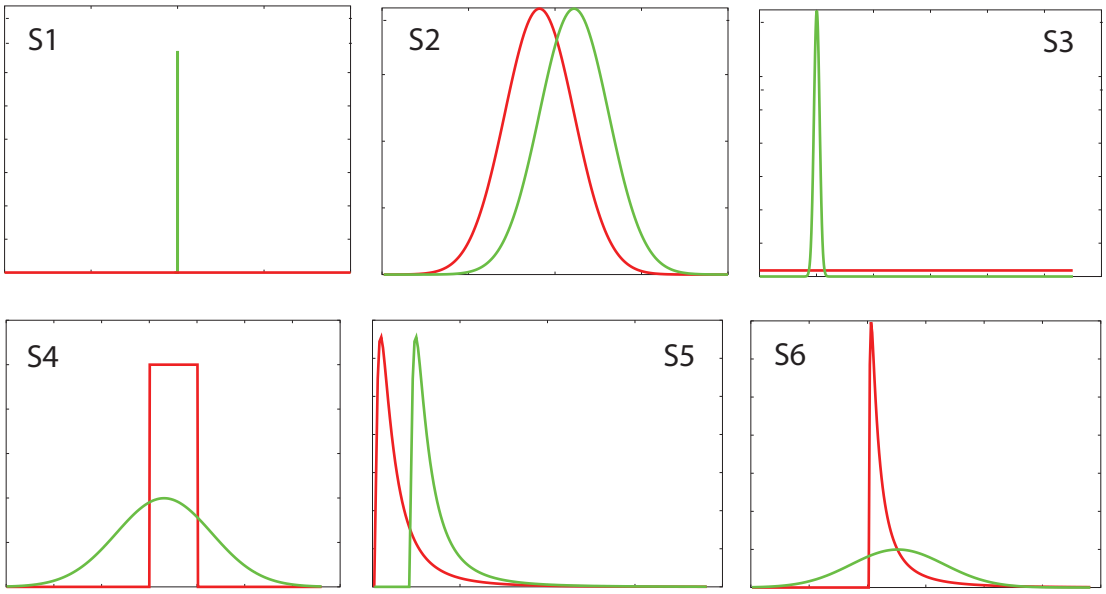


Figure 2: The six pairs of artificial distributions employed in the simulations.

NBS values of $100, 1000$, and 10000 with NTS values of 100 and 1000 to yield six different classification situations for each of the pairs S1 – S6. To reduce the influence of sampling noise on the results, we repeated simulations 100 times for each method for each of the six situations in each of the six pairs.

As a reference for method performance, we computed the optimal classification accuracy for each of the model pairs. Optimal performance is $99.9\%, 68.8\%, 97.2\%, 81.2\%, 88.1\%$, and 77.6% classification accuracy for pairs S1 – S6, respectively.

4.1 Results and Discussion

Two measures were employed to gauge the performance of the considered classification methods: classification accuracy and runtime. Since absolute runtime depends on the configuration of the simulating machine, we focused on the relative speed of the different methods by analyzing normalized runtimes. Both measures were recorded when boundary methods were estimating the number of boundaries to use (Sections 4.1.1 and 4.1.2) and when boundary methods were fixed to always use only a single boundary (Section 4.1.3).

4.1.1 Classification Accuracy

Table 2 shows classification accuracy for all methods across the six artificial distributions.

All binning methods yield comparatively poor classification accuracies and smoothed binning using a moving average of 9 is the worst of all considered methods. The poor accuracy of the binning methods arises from the difficulty to balance the control of noise and local discriminatory power within the binning method (i.e., setting the size of the bins and the smoothing window). For the 6 pairs of distributions considered here, smoothing with a moving average of 3 yielded the best performance of all binning methods: Bin-3 is the only binning method that, on average, does not perform worse than all other investigated methods.

E-search-1000 yielded the highest accuracies for all of the first 5 distribution pairs. The reason that it comes out second best on average is exclusively due to its poor performance on distribution pair S6. Accordingly, E-search-1000 achieved very good accuracies whenever the number of decision boundaries was estimated correctly. Highlighting the method's dependence on the number

Method	$S1$		$S2$		$S3$		$S4$		$S5$		$S6$		Avg.
	%	SE	%	SE	%	SE	%	SE	%	SE	%	SE	
Bin	71.46	2.00	66.82	0.48	80.64	9.45	76.94	2.02	81.63	2.91	60.65	1.46	73.03
Bin-3	86.86	5.99	68.39	0.18	91.96	2.79	75.89	2.72	76.90	5.91	59.87	1.89	76.64
Bin-5	81, 56	7.80	68.12	0.55	88.12	4.53	70.65	5.22	73.77	7.33	56.69	2.09	73.15
Bin-9	77.42	8.22	63.33	3.27	80.79	8.11	69.13	5.43	69.12	6.32	51.39	3.63	68.53
E-Search-10	87.73	0.84	68.65	0.10	94.12	0.11	74.19	0.37	66.34	4.38	56.52	0.92	74.59
E-Search-1000	99.73	0.07	68.97	0.16	96.96	0.11	80.79	0.24	87.80	0.17	63.63	0.15	82.98
A-Search-5	99.02	0.49	68.75	0.12	93.75	0.08	75.81	0.29	74.61	1.83	57.09	0.99	78.14
A-Search-10	99.72	0.14	68.91	0.16	96.94	0.08	79.56	0.09	83.34	1.19	60.89	1.02	81.56
Parametric	100	0	69.04	0.07	97.16	0.01	79.65	0.20	75.65	0.39	54.40	1.15	79.32
kernel	100	0	69.00	0.10	96.46	0.03	74.92	0.04	83.13	1.27	60.82	0.11	80.72
k-NN	98.98	0.49	67.42	0.13	96.65	0.27	79.87	0.22	87.74	0.09	76.12	0.27	84.46
Vicinity	95.00	0.36	68.26	0.56	96.42	0.01	79.57	0.13	86.66	0.25	71.55	1.11	82.91

Table 2: Classification accuracy and corresponding standard error for artificial distributions when methods chose the number of boundaries.

of potential decision boundaries that are considered, E-search-10 yielded comparatively poor accuracy. The low accuracy of E-search-10 is mainly due to this method's difficulties to deal with strongly peaked distributions over a comparatively wide range of sampled values (e.g., S1, S5).

The accuracies achieved by A-search-5 and A-search-10 indicate that a substantial reduction of potential boundaries is possible, if the number of potential boundaries considered in each step of the A-search is not too low. The A-search-10 performs as well as or nearly as well as the E-search-1000 across all 6 pairs of distributions. The A-search-5 performs worse than both the E-search-1000 and the A-search-10, but still better than the E-search-10.

Despite the normality assumption underlying the parametric method, it performs close to the optimum not only for the two pairs of normal distributions (S1, S2), but also for the pairs that involve uniform distributions (S3, S4). The method produces clearly suboptimal accuracies only for those pairs that involve log-normal distributions (S5, S6). Consequently, the parametric method can be quite robust against violations of its assumption, but only if the actual distributions are not too skewed.

The kernel method relies on the normality assumption to a lesser extent than the parametric method, because it considers all individual samples for classification. This renders the kernel method more robust against violations of the normal assumption than the parametric method (e.g., S5, S6). Nevertheless, the kernel method tends to perform sub-optimally as soon as one of the two involved distributions is non-normal (e.g., S4 – S6).

The k-NN achieved the best average accuracy of all considered methods. It performs as well or nearly as well as the best methods on distribution pairs S1 – S5 and clearly better than any other method on distribution pair S6. The k-NN outperforms all boundary methods on S6, because the k-NN needs not estimate the number of boundaries. The k-NN outperforms the other two direct methods, because (a) in contrast to the kernel method the k-NN makes no assumptions on the type of distributions and (b) in contrast to the vicinity method the k-NN considers a vicinity that has a flexible size. As a result, for the 6 considered distribution pairs, the k-NN exhibits the best balance between optimizing accuracies and being applicable across a range of different situations.

The vicinity search yielded slightly lower than optimal accuracies across all pairs, and clearly lower accuracies when the two distributions had very different kurtosis (e.g., S1, S6). These accuracy decrements arise from the method's use of a fixed fraction of the joint range that is covered by the two distributions. This renders the vicinity search prone to misclassification, if samples fall close to steeply rising densities.

4.1.2 Runtime

Each methods' normalized runtime is shown in Table 3: The methods differ considerably in how long it takes to apply them. By far the slowest method on average is the E-search 1000, which

Method	$S1$	$S2$	$S3$	$S4$	$S5$	$S6$	Average
Bin	6.1E-6(2.1E-6)	7.2E-6(2.8E-6)	6.0E-6(2.3E-6)	6.5E-6(2.3E-6)	5.5E-6(2.0E-6)	5.8E-6(2.0E-6)	6.2E-6
Bin-3	6.8E-6(2.6E-6)	8.1E-6(3.2E-6)	6.9E-6(2.7E-6)	7.2E-6(2.8E-6)	6.5E-6(2.5E-6)	6.5E-6(2.5E-6)	7.0E-6
Bin-5	6.9E-6(2.6E-6)	8.1E-6(3.2E-6)	6.9E-6(2.7E-6)	7.0E-6(2.8E-6)	6.4E-6(2.5E-6)	6.4E-6(2.5E-6)	7.0E-6
Bin-9	6.9E-6(2.6E-6)	8.0E-6(3.3E-6)	6.9E-6(2.7E-6)	7.0E-6(2.8E-6)	6.4E-6(2.5E-6)	6.4E-6(2.5E-6)	6.9E-6
E-Search-10	1.1E-4(5.4E-5)	2.2E-5(9.9E-6)	1.0E-4(4.8E-5)	1.1E-4(5.4E-5)	5.1E-5(2.4E-5)	1.9E-5(8.4E-6)	7.0E-5
E-Search-1000	1(4.9E-1)	3.4E-2(1.4E-2)	8.9E-1(4.4E-1)	9.9E-1(4.9E-1)	3.6E-1(1.7E-1)	2.5E-3(6.6E-4)	5.5E-1
A-Search-5	5.3E-4(2.7E-4)	8.7E-5(4.4E-5)	9.2E-5(4.28E-5)	9.9E-5(4.8E-5)	8.0E-5(3.9E-5)	5.4E-5(2.7E-5)	1.6E-4
A-Search-10	1.0E-3(5.3E-4)	8.8E-5(4.3E-5)	5.2E-4(2.6E-4)	5.0E-4(2.5E-4)	2.2E-4(1.0E-4)	7.2E-5(3.4E-5)	4.1E-4
Parametric	5.0E-6(1.8E-6)	5.1E-6(1.8E-6)	4.8E-6(1.7E-6)	5.3E-6(1.8E-6)	5.0E-6(1.8E-6)	5.0E-6(1.8E-6)	5.0E-6
kernel	1.8E-1(1.4E-1)	1.1E-1(7.9E-2)	1.2E-1(8.7E-2)	1.1E-1(8.0E-2)	1.1E-1(8.1E-2)	1.1E-1(8.0E-2)	1.2E-1
k-NN	3.7E-3(2.7E-3)	3.7E-3(2.7E-3)	3.7E-3(2.6E-3)	3.7E-3(2.7E-3)	3.7E-3(2.7E-3)	3.7E-3(2.6E-3)	3.7E-3
Vicinity	2.4E-3(1.7E-3)	2.2E-3(1.6E-3)	2.3E-3(1.7E-3)	2.2E-3(1.6E-3)	2.5E-3(1.8E-3)	2.5E-3(1.8E-3)	2.4E-3

Table 3: Normalized runtime for classification of artificial distributions when methods chose the number of boundaries. Standard error is shown in parentheses.

takes up to 3 seconds on those pairs that are estimated to require two boundaries (S1 and S3 – S5). These large runtimes are a result of the way the E-search treats pairs for which two boundaries need to be determined. In such situations, as explained in Section 3.1.3, the required time scales quadratically with the number of equidistant points in the E-search. If only a single boundary needs to be determined, the E-search-1000 is much faster, though still comparatively slow (see S2 and S6).

A similar pattern across the six distribution pairs is evident also for A-search-10, A-search-5, and E-search-10. However, all of these three are much faster than the E-search-1000 due to the fact that they need to consider much fewer equidistant points during boundary search. All other boundary methods, the binning variants and the parametric method, are very quick.

The direct methods are generally slower than the boundary methods, because for each new sample that needs to be classified, the direct methods need to consider all of the samples of the two distributions to reach a classification decision. As a result, the direct methods not only need more time than boundary methods, but also show a much steeper increase in runtime with an increase in the number of samples that need to be classified. The kernel method is the slowest of all direct methods, because it requires frequent evaluation of the Gaussian density function, which is a comparatively expensive computational process.

4.1.3 Single Boundary Results

We have already discussed the possible impact that the (incorrect) estimation of the number of decision boundaries can have on boundary methods (see Section 3.1). To quantify the effect the number of decision boundaries has on method performance, we conducted additional simulations that restrict boundary methods to the use of a single boundary.

Accuracy and runtime results of the single boundary simulations are displayed in Tables 4 and 5.[3] The figure indicates that restriction to use only a single boundary leads to a – for some methods marked – speed-up. For all methods, it also leads to a substantial decrease in accuracy in situations in which the two distributions are nested. Accordingly, allowing for more than a single boundary and estimating the number of boundaries seems (a) necessary if nested distributions can be expected to arise and (b) reasonable if incorrectly assuming two boundaries is either a rare case or does not lead to substantial increases in runtime.

4.1.4 Discussion

The results obtained for the six pairs of artificial distributions allow a first assessment of the suitability of the considered methods.

[3]The direct methods are not influenced by the restriction of using only a single boundary. The results of these methods are only displayed for reference and will not be discussed further.

Method	$S1$		$S2$		$S3$		$S4$		$S5$		$S6$		Avg.
	%	SE	%	SE	%	SE	%	SE	%	SE	%	SE	
Bin	58.58	2.09	67.70	0.32	71.09	7.12	62.20	0.67	81.86	3.10	61.13	1.16	67.09
Bin-3	68.33	2.98	68.79	0.18	81.13	0.69	60.26	1.46	77.02	5.90	59.57	2.02	69.18
Bin-5	65.39	4.13	68.30	0.48	77.56	1.50	58.47	2.28	73.81	7.49	56.61	2.10	66.69
Bin-9	63.17	3.99	63.37	3.27	71.01	5.28	57.74	2.19	69.14	6.27	51.36	3.70	62.63
E-Search-10	69.24	0.32	68.48	0.13	88.89	0.05	65.02	0.40	66.42	4.42	56.11	0.74	69.03
E-Search-1000	74.85	0.03	68.96	0.11	89.48	0.09	68.85	0.18	87.94	0.12	63.13	0.06	75.54
A-Search-5	74.38	0.11	68.70	0.20	88.98	0.07	68.35	0.15	86.71	0.17	56.79	0.84	73.98
A-Search-10	74.64	0.05	68.74	0.18	89.36	0.06	68.70	0.13	87.90	0.09	60.79	0.70	75.02
Parametric	50.14	0.04	50.04	0.03	64.53	0.52	53.45	0.18	51.49	0.61	50.36	1.20	53.33
kernel	100	0	69.20	0.12	96.49	0.03	74.80	0.11	83.13	1.26	60.77	0.05	80.73
k-NN	98.99	0.49	67.28	0.09	96.67	0.26	79.84	0.23	87.74	0.09	76.05	0.21	84.43
Vicinity	94.88	0.34	68.37	0.45	96.36	0.02	79.54	0.09	86.66	0.25	71.68	1.11	82.92

Table 4: Classification accuracy and corresponding standard error for artificial distributions when methods were restricted to use a single boundary

Method	$S1$	$S2$	$S3$	$S4$	$S5$	$S6$	Average
Bin	5.7E-6(2.1E-6)	6.8E-6(2.5E-6)	5.6E-6(2.2E-6)	5.9E-6(2.2E-6)	5.4E-6(2.0E-6)	5.5E-6(1.9E-6)	5.8E-6
Bin-3	6.5E-6(2.5E-6)	7.7E-6(3.0E-6)	6.5E-6(2.6E-6)	6.7E-6(2.6E-6)	6.2E-6(2.4E-6)	6.2E-6(2.3E-6)	6.6E-6
Bin-5	6.4E-6(2.6E-6)	7.7E-6(3.0E-6)	6.5E-6(2.6E-6)	6.7E-6(2.7E-6)	6.1E-6(2.4E-6)	6.2E-6(2.3E-6)	6.6E-6
Bin-9	6.4E-6(2.6E-6)	7.5E-6(3.1E-6)	6.4E-6(2.6E-6)	6.7E-6(2.7E-6)	6.2E-6(2.4E-6)	6.1E-6(2.4E-6)	6.5E-6
E-Search-10	1.9E-5(8.2E-6)	1.9E-5(8.6E-6)	2.5E-5(1.1E-5)	1.7E-5(7.9E-6)	1.8E-5(8.2E-6)	2.4E-5(1.1E-5)	2.0E-5
E-Search-1000	1.4E-3(6.7E-4)	1.8E-3(8.9E-4)	1.6E-3(7.8E-4)	1.4E-3(6.8E-4)	1.5E-3(7.0E-4)	1.5E-3(7.0E-4)	1.5E-3
A-Search-5	7.1E-5(3.5E-5)	6.6E-5(3.4E-5)	2.7E-5(1.3E-5)	4.8E-5(2.4E-5)	7.2E-5(3.6E-5)	5.3E-5(2.7E-5)	5.6E-5
A-Search-10	9.2E-5(4.4E-5)	9.8E-5(4.9E-5)	1.3E-4(7.1E-5)	9.6E-5(4.9E-5)	1.1E-4(5.2E-5)	7.0E-5(3.4E-5)	1.0E-4
Parametric	5.2E-6(1.8E-6)	4.5E-6(1.7E-6)	4.9E-6(1.7E-6)	5.2E-6(1.8E-6)	4.5E-6(1.7E-6)	4.9E-6(1.7E-6)	4.8E-6
kernel	2.0E-1(1.6E-1)	1.1E-1(8.0E-2)	1.2E-1(8.7E-2)	1.1E-1(8.1E-2)	1.1E-1(8.1E-2)	1.1E-1(7.9E-2)	1.3E-1
k-NN	3.7E-3(2.6E-3)	3.7E-3(2.6E-3)	3.7E-3(2.6E-3)	3.7E-3(2.6E-3)	3.7E-3(2.6E-3)	3.7E-3(2.6E-3)	3.7E-3
Vicinity	2.3E-3(1.7E-3)	2.1E-3(1.5E-3)	2.3E-3(1.6E-3)	2.2E-3(1.6E-3)	2.4E-3(1.7E-3)	2.5E-3(1.8E-3)	2.3E-3

Table 5: Normalized runtime for classification of artificial distributions when methods were restricted to use a single boundary. Standard error is shown in parentheses.

Although they are among the fastest methods, the low accuracy achieved by all binning methods and the difficulties in determining a proper bin size and smoothing window size makes them appear the least suitable of the considered methods.

The parametric method is the quickest of all considered methods and constitutes a very good choice, if the involved distributions can be approximated well by normal distributions. However, without prior knowledge about the types of distributions involved in classification, the parametric method seems to be a risky choice. The kernel method seems even less suitable suffering from the same problem but being slower than the parametric method.

The E-search-10 and the A-search-5 both suffer from low precision in searching for suitable boundaries: Both methods cannot rival their higher precision counterparts (E-search-1000 and A-search-10, respectively) in terms of accuracy. Although they are faster than their higher precision counterparts, the loss of accuracy is too big in most situations to justify the use of the E-search-10 or the A-search-5.

The vicinity search appears to be a reasonable method: It achieves good accuracies without requiring too much time. Nevertheless, it is outperformed by the only slightly slower k-NN on all 6 pairs of distributions.

The most promising methods are the E-search-1000, A-search-10, and the k-NN. The E-search-1000 achieves the highest accuracies in cases where the number of boundaries is estimated correctly, but is several orders of magnitude slower than the other two. The A-search-10 performs slightly worse than the E-search-1000, but is the quickest of all three methods. The k-NN also performed only slightly worse than the E-search-1000 while being considerably faster. In contrast to the search methods, (a) the runtime of the k-NN depends strongly on the number of samples that need to be classified and (b) its performance does not rely on successful estimation of the number of required decision boundaries.

Consequently, which method constitutes a good choice, depends on the properties of the situation at hand. If, besides accuracy, speed is of similar importance, the k-NN (if few samples have to be classified) or the A-search-10 (if many samples have to be classified) seem more suitable than the E-search-1000. If the distributions can be assumed to be approximately normal, the parametric method appears to be a viable choice. If distributions are of unknown type and / or there is a substantial chance of incorrectly estimating the number of required boundaries, the k-NN seems to be the most suitable method.

Against this background, an important question is what kind of (pairs of) distributions arise when the PBCM is applied to actual cognitive models. We now turn to this question by presenting and discussing simulations employing existing cognitive models.

5 Cognitive Models

In our second set of simulations, we applied the methods to distributions arising from 7 pairs of existing cognitive models.

For each pair of models A and B, we first generated two distributions $dist_A$ and $dist_B$ as described in Section 2. In a second step, 200 additional GOF differences were produced by generating 100 data sets from model A and 100 data sets from model B and fitting both models to these data sets. We will refer to these additional GOF differences as *test differences*. Each method was employed to classify the 200 test differences, given the two distributions $dist_A$ and $dist_B$. To gauge method performance, the number of times a method allowed to make a classification, classification accuracy, and runtime were measured.

For fitting the models, a variant of the Metropolis algorithm (e.g., Madras, 2002) was employed. For generating data from the models (both for establishing test differences and for establishing $dist_A$ and $dist_B$), model parameters were drawn randomly from the range of valid parameter values according to a uniform distribution (see Sections 5.1.1 – 5.1.3 for parameters and their ranges for the different model pairs).

Performance of the classification methods may vary widely across different modeling situations. To take this into account, we examined method performance for three different sets of models and, for each model pair, across 54 different modeling situations. The model sets were chosen to cover a range of different types of models: The first set comprises three mathematical models in the domain of memory retention (Section 5.1.1). The second set comprises two symbolic models in the domain of artificial grammar learning (Section 5.1.2). The third set comprises three connectionist models in the domain of perceptual choice (Section 5.1.3). The 54 different modeling situations were created by varying four factors that we assumed to vary across actual modeling situations. The first factor is the tightness of fit: We operationalized this factor by varying the number of swaps used in the Metropolis algorithm: $10, 100$, or 1000. The second factor also has three levels and concerns how many data points are available in each generated data set (NDP). The operationalization of this factor is model set-dependent and will be discussed when describing each model set. The third factor, called *noise level* (NL) controls how noisy the generated data are. As NDP, NL has three levels that are model set-specific. The fourth factor, NBS, concerns the number of samples that constitute distributions $dist_A$ and $dist_B$. Our simulations employed $NBS = 100$ and $NBS = 1000$.

5.1 Model Sets

5.1.1 Memory Models

The three memory models we employed ($M1, M2$, and $M3$, see also Pitt & Myung, 2002) predict the probability of recalling any of a set of learned items after time t according to the following

formulas:

$$M1 : (1+t)^{-a}, a \in [0,2],$$
$$M2 : (b+t)^{-a}, a \in [0,2], b \in [1,2],$$
$$M3 : (1+bt)^{-a}, a \in [0,2], b \in [0,2],$$

where a and b are free model parameters the value of which was restricted to the corresponding intervals during simulations. These three models yield three pairs of models, $(M1, M2)$, $(M1, M3)$, and $(M2, M3)$. Each method was applied to each of these three pairs across all of the 54 modeling situations mentioned above.

To generate data from any of the models, the model was first used to predict recall probabilities for a number of times, $t_1, \ldots, t_{NDP}$. These times were equidistantly distributed across the interval $[0.1, 8.1]$ using three different levels of number of times: $NDP = 5, NDP = 20$, and $NDP = 100$. The obtained probabilities served as the basis for sampling the number of recalled items from a binomial distribution for each of the NDP-many times. The maximum number of times that could be recalled was also varied in three levels to manipulate the amount of noise in the generated data. We used $NL = 10, NL = 100$, and $NL = 1000$ to simulate high noise, moderate noise, and low noise data sets, respectively. Models were then fit to the noisy data sets to generate GOF differences for building $dist_A$ and $dist_B$ as well as the test differences.

5.1.2 Artificial Grammar Learning Models

The symbolic model pair was used to simulate learning of letter strings from four lists of strings employed in Miller (1958). Each of the four lists contained 9 strings consisting of at least 4 and at maximum 7 letters from the set {S, X, G, N}. For two of these lists the strings were randomly assembled. The other two lists contained strings that were constructed from an artificial grammar and, thus, contained some regularities.

The models had to learn the strings from the lists by being repeatedly presented all strings from a list. After each complete presentation of a list, the number of strings the model had learned was determined. If, for example, a random list and a grammar list were presented 10 times each, this yielded 20 data points: 10 numbers of learned items for each of the two lists. To vary how many data points are available, we manipulated the number of lists and the number of repetitions: $NDP = 4$ was realized by using one grammar list and one random list with two repetitions each; $NDP = 20$ was realized by using one grammar list and one random list with 10 repetitions each; $NDP = 60$ was realized by using both grammar lists and both random lists with 15 repetitions each. The amount of noise was varied by manipulating the number of model runs we averaged across: $NL = 5, NL = 30$, and $NL = 100$ runs were used for the high, moderate, and low noise conditions, respectively.

The two models that we considered are the competitive chunking model (CC) proposed by Servan-Schreiber and Anderson (1990) and the PARSER model (Perruchet & Vinter, 1998; Perruchet & Pacton, 2006). The CC model has two free parameters, a competition parameter c and a decay parameter d. In our simulations, c was from the interval $(0, 3)$ and d was from the interval $(0, 1)$. The PARSER model has four parameters, the learning rate m, a decay parameter f, an interference parameter i, and a threshold t. Parameter values were from the interval $(0, 2)$, $(0, 1)$, $(0, 0.5)$, and $(0, 3)$ for m, f, i, and t, respectively.

5.1.3 Perceptual Choice Models

The third model set comprised three connectionist models, all of which were variants of the *leaky, competing accumulator* (LCA) model proposed by (Usher & McClelland, 2001). The LCA consists of a set of units that receive sensory information from the environment. The units represent different conflicting interpretations of an available stimulus and the environmental input represents the perceptual evidence for the different interpretations. The activation of the units in the LCA is

not only influenced by the input but also by mutually inhibitory connections between the units. The change in activation of each unit is determined as follows:

$$c_i'(t) = \left(inp_i(t) - leak * c_i(t) - inh * \sum_{k \neq i} c_k(t) \right) * \frac{h}{\tau} + \xi * \sqrt{\frac{h}{\tau}},$$

where $c_i'(t)$ is the change of activation in unit i at time t, $c_i(t)$ is the activation in unit i at time t, $inp_i(t)$ is perceptual evidence at time t, $leak$ is a factor which determines how quickly a unit's activation will decrease to its resting level when $inp_i(t)$ is zero, inh is the strength of the inhibitory connections between units, h is the step-size used for numerically solving the differential equation, τ is a time-scale factor, and xi is a Gaussian noise term with mean zero. If the activation of one of the units grows beyond a certain threshold t, processing stops. The selected interpretation is assumed to be that interpretation, which is represented by the unit that exceeded the threshold.

In the simulations, inp_i was from the interval $[0, 1]$ subject to the constraint that $\sum_i inp_i = 1$, $leak$ was from the interval $[0, 1]$, inh was from the interval $[0, 2]$, the standard deviation of the noise term ξ was from the interval $[0, 1]$, t was from the interval $[0, 3]$, and $h = \tau = 100$ were fixed.

The LCA was complemented by two additional models that arise from slight modifications of the LCA. The first modification eliminated any inhibitory connections between units ($inh = 0$) and the second modification assumed no leakage ($leak = 0$). We term these models the *leaky accumulator* (LA) model and the *competing accumulator* (CA) model, respectively. The models were used to simulate perceptual choice in the task employed by (Vickers, 1970). We ran the models multiple times in each of the task's 6 conditions to obtain corresponding response time distributions. Sets of quantiles of the distributions served as the data points when generating GOF difference values.

To vary the number of data points, we manipulated the number of quantiles that were computed from the distributions. For each condition, we either considered only the median (yielding $NDP = 6$) or the quartiles (yielding $NDP = 18$) or the 10%-quantiles (yielding $NDP = 54$). The level of noise was controlled by the number of model runs: $NL = 150$, $NL = 1500$, and $NL = 15000$ runs were used for realizing levels of high, moderate, and low noise, respectively.

5.2 Results and Discussion

Simulation of existing cognitive models allowed assessing (a) our hypotheses on distributions arising in the scope of the PBCM and (b) the performance of the classification methods on actual model-generated distributions. We first consider distribution properties in Section 5.2.1 before examining method performance on the different model sets in Sections 5.2.2 – 5.2.5.

5.2.1 Distribution Properties

Because the distributions from situations with $NBS = 1000$ allow for more reliably assessing distribution properties, we restricted testing our hypotheses on distributions to these situations (27 of the 54 situations for each model pair). To further increase reliability, we combined the 1000 samples of each distribution with the 100 test differences generated for each distribution in each pair. Given the 7 model pairs considered in our simulations, this yielded $7 * 27 = 189$ pairs of distributions, where each distribution comprised 1100 GOF differences.

Hypotheses 1 and 2 The first two hypotheses (see inequalities (1) and (2) in Section 2.1) state that each model tends to fit its own data at least as well as the other model's data. To examine these hypotheses, we conducted one-tailed Mann-Whitney-U tests (Mann & Whitney, 1947) that pitted our hypotheses against the alternative that a model tended to fit its own data worse than the other model's data. This involved 189 U tests for each of the two hypotheses: For each model of each

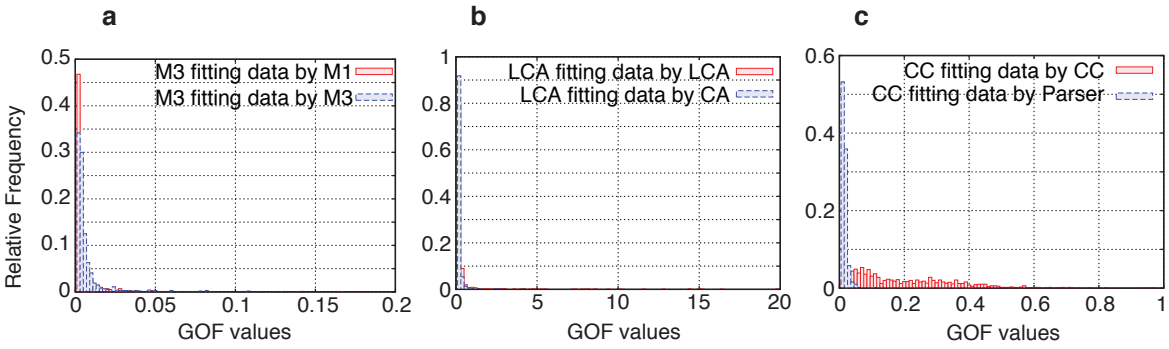


Figure 3: Distribution of GOF values (Mean Squared Error) when fitting a model's own data and when fitting a competing model's data. a) Memory model M3 fitting its own data and fitting M1's data; b) LCA fitting its own data and fitting CA's data; c) CC fitting its own data and fitting Parser's data.

model pair a test was conducted for each of the 27 situations. Given the large number of tests, the level of significance, α, was Bonferroni-corrected to $\alpha = 0.00026 \approx 0.05/189$.

For 17 and 9 pairs of distributions we found a statistically significant violation of our first and second hypothesis, respectively. This means that the distributions obtained from the simulation of existing cognitive models contradicted our assumption that a model tends to fit its own data better than the other model's data in 26 of 378 cases (i.e., in 6.9% of all cases). Of these 26 cases, 9 arose from the pair (M1, M3) of the memory model set such that in all of these cases M3 tended to fit its own data worse than the data of M1. This tendency was not very strong in the sense that the percentage of GOF values, for which $gof_{M3}^{\mathbf{x}_{M1}} > gof_{M3}^{\mathbf{x}_{M3}}$ held, remained below 58% for all 9 cases (see Figure 3a for an illustration of the most extreme case).

Another 4 cases violating our first hypothesis arose from the pair (LCA, CA) of the perceptual choice model set: The LCA model tended to fit its own data worse than the data of the CA. Again this tendency was not very strong yielding between 55% and 63% of GOF values, for which $gof_{LCA}^{\mathbf{x}_{CA}} > gof_{LCA}^{\mathbf{x}_{LCA}}$ held (see Figure 3b for an illustration of the most extreme case).

The remaining 13 cases contradicting our first hypothesis arose from the artificial grammar learning model pair (CC, PARSER): The CC model tended to fit its own data worse than the data of the PARSER model. This tendency was much stronger than in the other two model pairs. The percentage of GOF values, for which $gof_{CC}^{\mathbf{x}_{PARSER}} > gof_{CC}^{\mathbf{x}_{CC}}$ held, were close to 90% with the highest amounting to 95% (see Figure 3c for an illustration of this case).

In sum, we found few violations of the first two hypotheses and all of the few cases that were evident arose from situations in which a more complex model generated data that was harder to fit for itself than the data of the other (simpler) model. Thus, our simulations suggest that the first two hypotheses hold most of the time, and that if they do not hold, it is for the more complex model.

Hypothesis 3 The procedure to check the third hypothesis (see inequality (3) in Section 2.1) was very similar to the procedure employed for the first two hypotheses. We conducted one-tailed Mann-Whitney-U tests (Mann & Whitney, 1947) that tested our hypotheses against the alternative that GOF differences obtained on data generated by the first model in each model pair tended to be smaller than GOF differences obtained on data generated by the second model in each model pair.

The resulting 189 U tests found no evidence of a violation of our third hypothesis (all ps > 0.65; 180 ps > 0.999). This indicates that the relation of the bulk of the GOF difference distributions arising from the model pairs employed in our simulations is as expected based on the considerations in Section 2.1.

Hypotheses 4 and 5 That the relation of the bulk of the distributions follows the expectations formulated in hypothesis 3 does not rule out that distributions may be nested. Even if, for example,

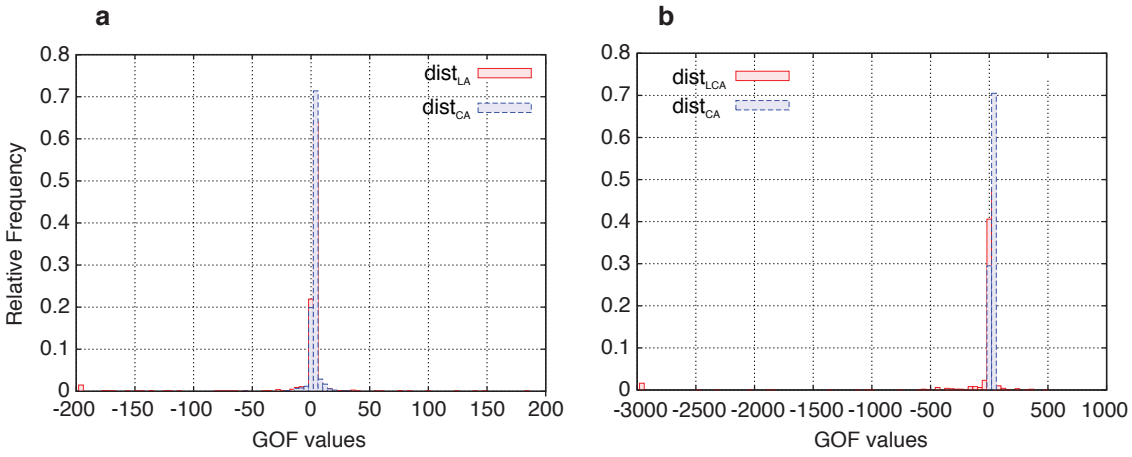


Figure 4: Nested pairs of GOF difference distributions for perceptual choice models. a) CA nested in LA; b) CA nested in LCA

the bulk of $dist_A$ lies to the right of $dist_B$, it is possible that regions left of the bulk of $dist_B$ exist where the probability density of $dist_A$ is higher than the density of $dist_B$. Hypotheses 4 and 5 (see inequalities (1) and (2) in Section 2.1) formulate the expectation that such situations should rarely (if ever) occur.

To check whether the GOF difference distributions from our simulations contradicted the last two hypotheses we proceeded as follows. First we investigated for all 189 pairs of difference distributions whether the distributions appeared to be nested, that is, whether the range of one of the distributions completely spanned the range of the other distribution: 67 of the 189 distribution pairs were found to be nested. For each of these nested pairs only one tail of the nesting distribution potentially contradicts our last two hypotheses. If $dist_A$ / $dist_B$ is the nesting distribution, only the left / right tail constitute a possible contradiction to our hypotheses.

In the second step, we computed the number of GOF differences that constituted a possible contradiction to our hypotheses. Across the 67 nested pairs the number of critical GOF differences ranged from 1 to 29 with a mean of 6.06 and a standard deviation of 7.99.

Third, we investigated which of these number of critical differences was statistically reliable. To do so, we estimated the probability of obtaining GOF differences from the nesting distribution that are more extreme than the most extreme value of the nested distribution. If, for example, $dist_A$ was the nesting distribution and 2 GOF differences from $dist_A$ were smaller than the smallest GOF difference from $dist_B$, we estimated the probability to be $\frac{2}{1100}$. If hypothesis 4 were correct, the probability of obtaining a GOF difference from $dist_B$, which is smaller than the actually observed smallest difference from $dist_B$, is at least as high as the probability estimated for $dist_A$. Accordingly, the probability to observe no GOF differences from $dist_B$ in this range can be approximated as $(1 - \frac{2}{1100})^{1100} \approx 0.135$. In this way, for all nested pairs, we computed the probability of not observing GOF differences in the critical range from the nested distribution assuming that our hypotheses hold. A Bonferroni-corrected $\alpha = 0.05/67 \approx 0.00075$ was used to determine whether a nested pair reliably contradicted one of our last two hypotheses.

Using this criterion, 15 of the 67 nested pairs constituted statistically significant contradictions to either hypothesis 4 or hypothesis 5. For these 15 pairs, the number of critical GOF differences ranged from 9 to 29 with a mean of 19.6 and a standard deviation of 6.21. All of the 15 pairs arose from the perceptual choice model set: 8 arose from the pair (LA, CA) and 7 from the pair (LCA, CA). The histogram of the most extreme case for pair (LA, CA), 29 critical GOF differences, is shown in Figure 4a. The histograms of the most extreme cases are shown in Figure 4.

The analyses of several hundred pairs of distributions arising from our simulations lent support

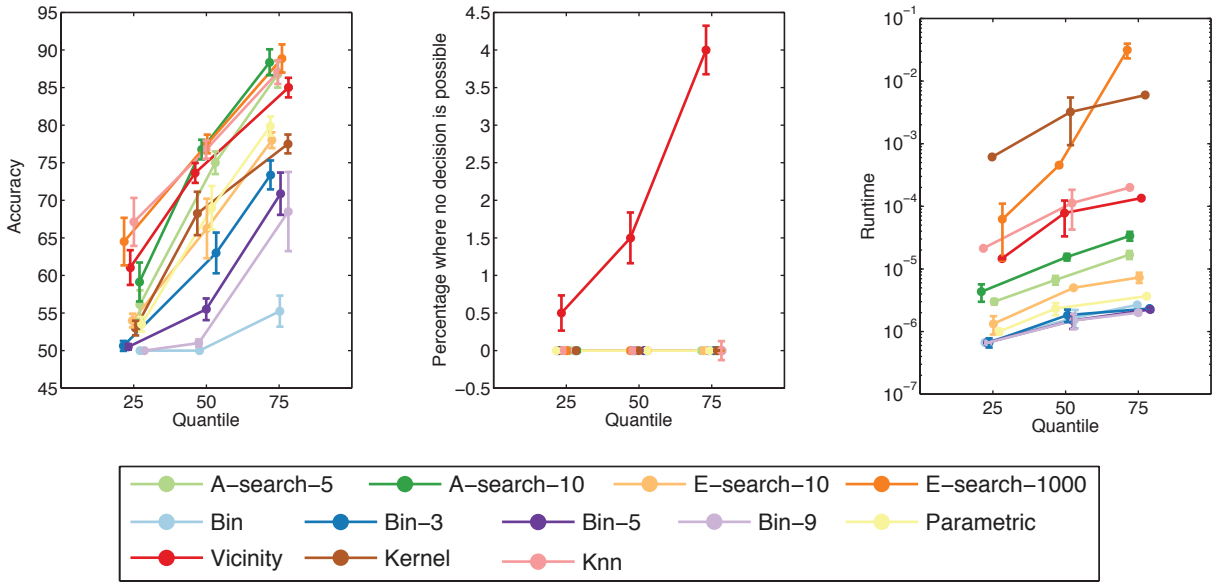


Figure 5: Results for memory retention models.

to the 5 hypotheses formalized by inequalities (1) – (5) (see Section 2.1). Although some of the distribution pairs contradicted hypotheses 1, 2, 4, and 5, such contradicting pairs were rare (7% and 8% for hypotheses $1, 2$ and $4, 5$, respectively) and deviations from the expected distribution properties was generally weak. Accordingly our analyses suggest two main conclusions: First, the by far most common relation between $dist_A$ and $dist_B$ is the one shown in the left panel of Figure 1. Second, even if distributions are nested, ignoring this nestedness will not strongly impact classification accuracy, because only very few GOF differences from the nesting distribution lie in the critical region.

5.2.2 Memory Retention

To assess the methods' performance we computed, for each method, model pair, and situation, (a) the percentage of correct classification for cases when a classification was possible, (b) the percentage of CS cases, and (c) normalized runtime.

For each of the three measures, the first, second (median), and third quartiles (and associated standard errors) were determined (a) for each method and model pair across all situations as well as (b) for each method across all model pairs and situations. Since method performance was similar across the different model pairs we present and discuss results collapsed across pairs in this section. Results for each model pair individually are provided in Appendix A.

Figure 5 displays the quartiles of the classification accuracy, percentage of CS cases, and runtime for the different methods. The vicinity search method is the only one that exhibits a noticeable, though small, number of CS cases. Note, however, that the error bar associated with the third quartile of the k-NN indicates that this method also occasionally yielded a few CS cases. All other methods did not encounter any situations in which they were unable to classify the test differences.

Normalized performance varied considerably across methods. All binning methods perform worse than all other methods. This is particularly clear for Bin, Bin-5, and Bin-9; it is less extreme but still noticeable for Bin-3. The next best performing group of methods are the kernel method, the parametric method, and the E-search-10. All three of these perform on a similar level: Clearly better than the binning methods, but also clearly worse than performance of the remaining 5 methods. These 5 methods, the vicinity search, the k-NN, the A-search-5, the A-search-10, and the E-search-1000 all show similar performance for the median, and the third quartile. More

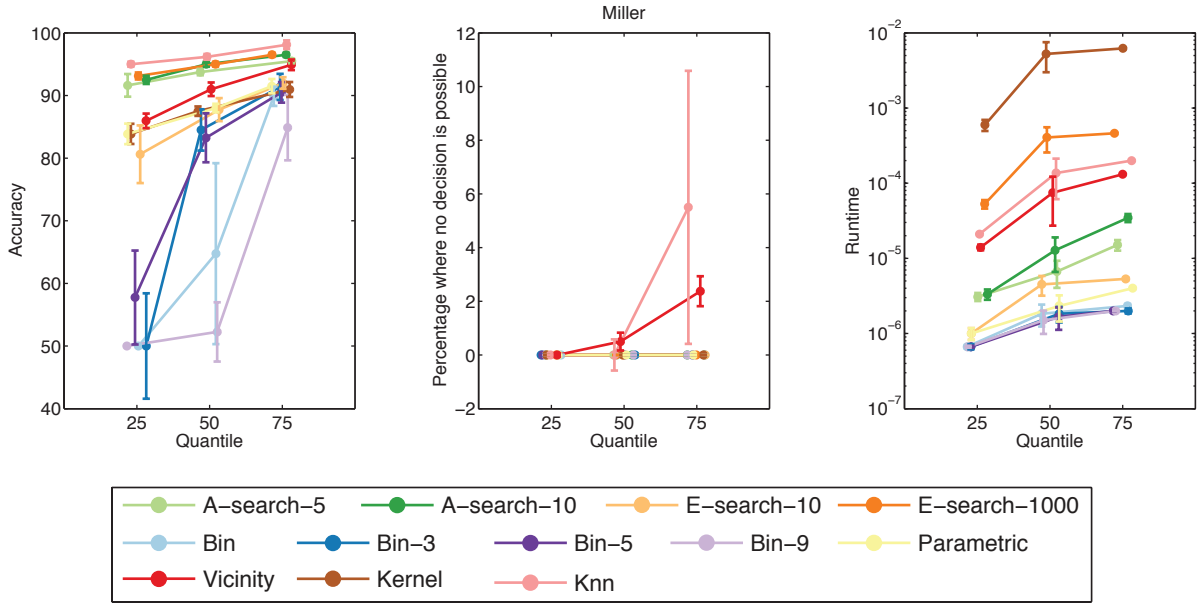


Figure 6: Results for artificial grammar learning models.

marked differences mainly exist for the first quartile: The vicinity search, the A-search-10, and, in particular, the A-search-5, perform worse than the k-NN or the E-search-1000. The reason for the performance drop in some situations is likely due to the chance of getting stuck in local minima (A-search, see Section 3.3) and the fixed size of the vicinity (see Sections 3.3 and 4.1.1). Consequently, the E-search-1000 and the k-NN performed best of all methods across the three memory retention model pairs and across the different situations.

Regarding runtime, the boundary methods generally require less time for classification than the direct methods. The only exception in our simulations was the E-search-1000: The third runtime quartile for this method was many times higher than for any other method. This high value results from the quadratic runtime complexity of the E-search when distributions are assumed to be nested (17 pairs from the memory models were nested, see Section 5.2.1). If only a single boundary had to be estimated, the E-search-1000 was much faster (see 1^{st} and 2^{nd} quartile) though still slower than any other boundary method. The second and third slowest boundary methods are the A-search-10 and the A-search-5, respectively. All other boundary methods are extremely quick. Of the direct methods, the k-NN and vicinity search are much faster than the kernel method, but still a bit slower than the boundary methods.

5.2.3 Artificial Grammar Learning

Classification accuracy, percentage of CS cases, and runtime for each method on the artificial grammar learning models are shown in Figure 6.

The results largely mirrored those obtained for the memory model pairs. Vicinity search and k-NN were the only methods that occasionally yielded a small number of CS cases. Classification accuracy was worst for all binning methods, medium for the kernel method, the parametric method, and the E-search-10, and best for the E-search-1000, the k-NN, and the two A-search variants. E-search-1000 was again the slowest boundary method and the only boundary method that was not faster than all direct methods. The slowest direct method was the kernel method, followed by the k-NN and the vicinity search. In comparison to the memory models, runtime of the E- and A-search methods tended to be lower and less variable across situations, because only two of the distribution pairs arising from the artificial grammar learning models were assumed to be nested.

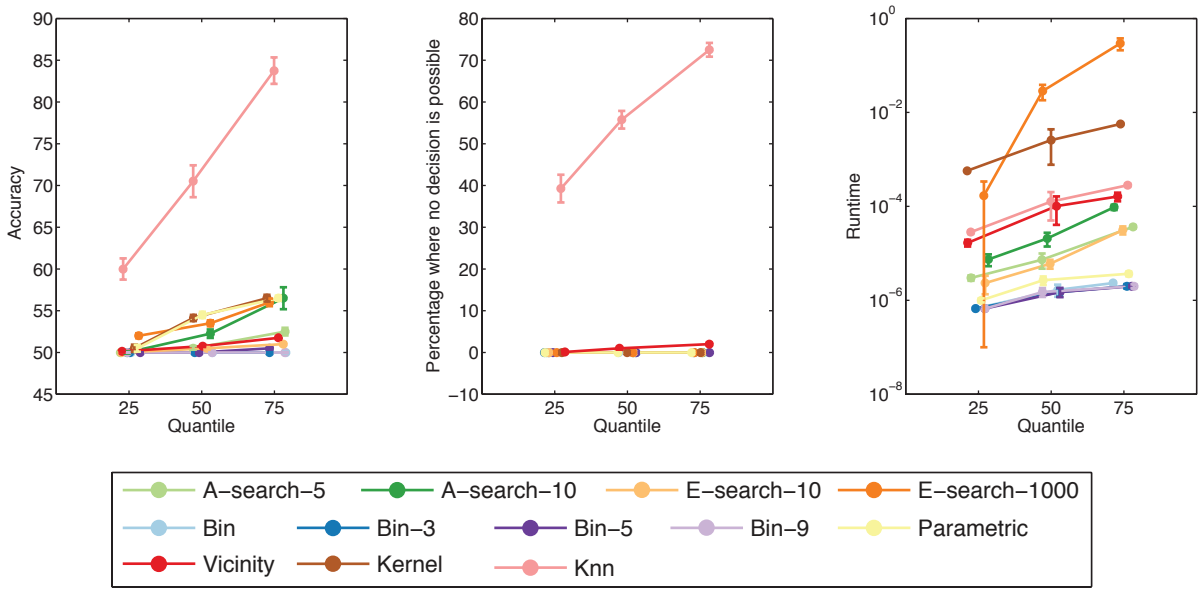


Figure 7: Results for perceptual choice models.

5.2.4 Perceptual Choice

The classification accuracy, percentage of CS cases, and runtime of each method collapsed across model pairs[4] and situations for the perceptual choice models are shown in Figure 7.

As for the memory models and the artificial grammar learning models, vicinity search encountered a few CS cases, while the kernel method and all boundary methods allowed for a classification decision in all situations. However, in contrast to the previous model sets, the k-NN exhibited a large number of CS cases for the perceptual choice models: In many situations, more than half of all test differences could not be classified by the k-NN and for a substantial fraction of all situations more than 70% of the test differences constituted CS cases. Interestingly, this large percentage of CS cases of the k-NN was mirrored by the classification accuracy of this method in the sense that the k-NN was the only method that clearly and consistently performed above chance. All other methods were either on or very close to chance level across most situations (all binning methods, the E-search-10, vicinity search, A-search-5) or only slightly reliably above chance for some of the considered situations (E-search-1000, the parametric method, the kernel method, A-search-10).

The reason for this performance pattern becomes clear when considering the nature of the distributions arising from the perceptual choice models. As for the distribution pair shown in Figure 4b, for most pairs, a substantial number of GOF differences from both distributions and of the test differences were equal to zero. For the k-NN this high likelihood of zero differences under both distributions meant that the $k = 10$ nearest neighbors for a test difference of zero were all zero for both distributions and, hence, the k-NN was unable to classify any test difference of zero. In contrast, due to the way the other methods work (see Section 3), they often classified test differences that were zero. Consequently, on those test differences that were inherently hard to classify, the k-NN "refused" to take a decision, whereas all boundary methods proceeded with an – often incorrect – classification.

The runtime results followed the same pattern as the results obtained from the memory model simulations. The E-search-1000 was the slowest and the other search methods also exhibited considerable variation across the different situations. As for the memory models, this variation across situations stemmed from many situations that were found to be nested (Section 5.2.1). The runtime

[4]As for the memory models, method performance was very similar across model pairs. Therefore we decided to discuss results collapsed across model pairs. Results for each individual model pair are provided in Appendix A

of the direct methods again places them above all boundary methods except the E-search-1000, with the kernel method being slowest, the vicinity search being fastest, and the k-NN being slightly slower than the vicinity search, but faster than the kernel method.

5.2.5 Single Boundary Results

To investigate to what extent the usage of more than a single boundary increases or decreases classification performance on distribution pairs arising from actual cognitive models, we applied all boundary methods restricting them to the use of a single boundary to all of the 7 pairs of models.

The single boundary results were virtually identical to the multi-boundary results described above. Given the great similarity between the two sets of results, we only highlight the few notable differences in this section. Plots of the single boundary results for each model pair can be found in Appendix B.

The most pronounced difference concerns the runtime of the E-search-1000 for the memory models and the perceptual choice models. In the single boundary simulations, runtime of the E-search-1000 is much lower than for the multi boundary simulations. This difference is due to the fact that the costly estimation of two boundaries for the memory and perceptual choice model pairs that were found to be nested (see Section 5.2.1) was avoided. A similar but less extreme reduction was observable for the other search methods.

A second interesting finding is that both A-search methods achieved slightly better normalized performance for the perceptual choice model pairs: The third quartile of the A-search-10 / A-search-5 is now higher / as high as the third quartile of the E-search-1000. This suggests that for this particular set of models the "local optima" problem of the A-search was aggravated when more than one boundary had to be estimated.

In sum, there was little evidence that multi boundary and single boundary versions of the methods differ in terms of their classification performance. However, the runtime of all boundary methods was – sometimes considerably – reduced when using only a single boundary.

5.2.6 Discussion

The methods' performance characteristics observed in our cognitive modeling simulations were in line with and corroborated the observations made in the scope of the artificial distribution simulations.

Regarding classification performance, all binning methods were worse than all other methods. The E-search-10, the parametric method and the kernel method consistently performed better than the binning methods, but also often worse than the remaining 5 methods. The vicinity search and the A-search-5 often achieved good classification performance thus often outperforming the already mentioned methods. Nevertheless they fell short of the best three methods for both the artificial grammar learning methods and the perceptual choice models.

As for the artificial distributions, the E-search-1000, the k-NN, and the A-search-10 provided the best classification performance across all 7 model pairs. For the memory models, these three models were on par, but for the perceptual choice models and the artificial grammar learning models the k-NN consistently and reliably achieved the best classification accuracy and the E-search-1000 was slightly better than the A-search-10. The case of the perceptual choice models indicates that the k-NN was the only method that successfully "restricted" its classification decisions to those cases where an above-chance classification was possible. In comparison to the k-NN and the E-search-1000, the A-search-10 performed less robustly across different situations, showing a steeper increase from the first to the third quartile. However, the A-search-10 was many times faster than both the k-NN and the E-search-1000 and the k-NN was many times faster than the E-search-1000.

Direct analyses of GOF (difference) distributions corroborated our hypotheses that GOF difference distributions tend to be non-nested (see left panel of Figure 1). In line with this, the boundary methods performed virtually identical across the 7 considered model pairs when restricted to using

only a single boundary. Given that single boundary estimation is associated with – sometimes considerably – reduced runtimes for all boundary methods, our simulation results suggest that single boundary estimation is to be preferred over multi boundary estimation.

6 General Discussion

Although the performance differences between the candidate methods is of main interest to identifying a suitable classification approach for the PBCM, it is also interesting to consider the absolute performance achieved across our simulations. An optimal classification accuracy of 100% was rarely achieved: Only few methods yielded perfect classification performance in only few situations, and for the perceptual choice model set almost all methods performed close to chance (Figure 7).

Classification problems that cannot be solved perfectly are not uncommon (see, e.g., Duda et al., 2001), and our artificial distributions (Section 4) constitute specific examples of such problems. Nevertheless, the classification performance achieved on the actual cognitive model pairs raises the question whether (a) the considered models are inherently difficult to distinguish from each other or (b) the treatment of the model pairs by the PBCM gives rise to unnecessarily difficult classification problems.

The simulations reported in this article do not allow addressing this question, because they provide no information on the distinguishability of the models, which is independent of the PBCM. However, in a related set of simulations also employing the memory retention models (Section 5.1.1), Schultheis et al. (2013) found that the PBCM may occasionally be outperformed by other model comparison and selection techniques: Despite employing the classification methods that performed best in the simulations presented here, the PBCM yielded lower model recovery accuracy than, for example, the simple hold out method. Thus, it seems that the PBCM may, in certain situations, give rise to an unnecessarily complex classification problem in the sense that even a good solution to the problem leads to sub-optimal model comparison and selection performance.

On the other hand, the same study has shown that the PBCM is often superior to employing the straightforward GOF measure for model selection. By identifying easy to use and well performing classification methods, the work presented in this paper paves the way for appropriately employing the PBCM to obtain such superior performance.

7 Conclusion

Although it will depend on situational details which method should ideally be applied, the k-NN appears to constitute a good general choice for solving the classification problem associated with applying the PBCM. Of all considered methods, the k-NN performed best across the 6 artificial distribution pairs as well as across the 7 pairs of actual cognitive models. Furthermore, the k-NN, being a direct method, can, in principle, deal with situations requiring an arbitrary number of boundaries without the necessity to make any (incorrect) assumptions on the required number of boundaries. In addition, the k-NN was the only method that successfully avoided classifying test differences that could not be classified substantially above chance. A potential drawback of the k-NN is its runtime, which scales with the number of test differences that need to be classified. If a large number of differences need to be classified and the time required for classification is critical (e.g., in the scope of simulation studies such as the one of Cohen, Sanborn, & Shiffrin, 2008), other methods that work more quickly may be preferable. Of the quicker methods, the A-search-10 constitutes an appealing general choice: It is many times quicker than the k-NN and was also among the three top performing methods of those considered. If more specific information on the nature of the involved distributions is available, other methods may be preferable over both the k-NN or the A-search-10. For example, if the involved distributions are known to be normal or close to being so, the parametric method provides a very quick and accurate solution.

Our simulations also support the conclusion that the relation between distributions arising in the scope of the PBCM are not arbitrary: The obtained GOF difference distributions tended to be non-nested such that the bulk of one of the distributions lies to the right of the bulk of the other distribution. Whether there are any constraints on the shape of the difference distributions, however, remains an open question. While some model pairs (perceptual choice models) gave rise to distributions that could be reasonably approximated by normal distributions, other model pairs produced clearly non-normal distributions (memory models and artificial grammar learning models). Future work is required to more closely examine possible relations between the properties of the involved models and the properties of the GOF difference distributions arising from these models. Another interesting topic for further research concerns the question how differences in a-priori likelihood of the compared models (e.g., one model is more established) could be taken into account during classification.

Acknowledgments

The authors gratefully acknowledge support by the German Academic Exchange Service (DAAD) and the German Research Foundation (DFG) through the project R1-[ImageSpace], SFB/TR 8 Spatial Cognition. We are grateful to Rick Cooper and an anonymous reviewer for their suggestions for revising the manuscript. We thank Damoon Emami for his help in preparing the figures.

References

Cohen, A. L., Rotello, C. M., & MacMillan, N. A. (2008). Evaluating models of remember–know judgments: Complexity, mimicry, and discriminability. *Psychonomic Bulletin & Review*, *15*, 906-926.

Cohen, A. L., Sanborn, A. N., & Shiffrin, R. M. (2008). Model evaluation using grouped or individual data. *Psychonomic Bulletin & Review*, *15*, 692-712.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification*. New York: Wiley.

Jang, Y., Wixted, J. T., & Huber, D. E. (2011). The diagnosticity of individual data for model selection: Comparing signal-detection models of recognition memory. *Psychonomic Bulletin & Review*, *18*, 751-757.

Madras, N. N. (2002). *Lectures on monte carlo methods*. Providence, Rhode Island: American Mathematical Society.

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, *18*(1), 50 – 60.

Miller, G. A. (1958). Free recall of redundant strings of letters. *Journal of Experimental Psychology*, *56*, 485 – 491.

Perea, M., Gomez, P., & Fraga, I. (2010). Masked nonword repetition effects in yes/no and go/no-go lexical decision: A test of the evidence accumulation and deadline accounts. *Psychon B & Rev*, *17*, 369-374.

Perruchet, P., & Pacton, S. (2006). Implicit learning and statistical learning: one phenomenon, two approaches. *Trends in Cognitive Sciences*, *10*, 233 – 238.

Perruchet, P., & Vinter, A. (1998). Parser: A model for word segmentation. *Journal of Memory and Language*, *39*, 246 – 263.

Pitt, M. A., & Myung, J. (2002). When a good fit can be bad. *Trends in Cognitive Sciences*, *6*, 421-425.

Roberts, S., & Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. *Psychological Review*, *107*, 358 - 367.

Schultheis, H., & Naidu, P. (2014). Multi-model comparison using the cross-fitting method. In P. Bello, M. Guarini, M. McShane, & B. Scassellati (Eds.), *Proceedings of the 36th annual*

conference of the Cognitive Science Society (pp. 1389 – 1394). Austin, TX: Cognitive Science Society.

Schultheis, H., Singhaniya, A., & Chaplot, D. (2013). Comparing model comparison methods. In M. Knauff, M. Pauen, N. Sebanz, & I. Wachsmuth (Eds.), *Proceedings of the 35th annual conference of the Cognitive Science Society.* Austin, TX: Cognitive Science Society.

Servan-Schreiber, E., & Anderson, J. R. (1990). Learning artificial grammars with competitive chunking. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *16*, 592 – 608.

Shiffrin, R. M., Lee, M. D., Kim, W., & Wagenmakers, E.-J. (2008). A survey of model evaluation approaches with a tutorial on hierarchical bayesian methods. *Cognitive Science*, *32*, 1248-1284.

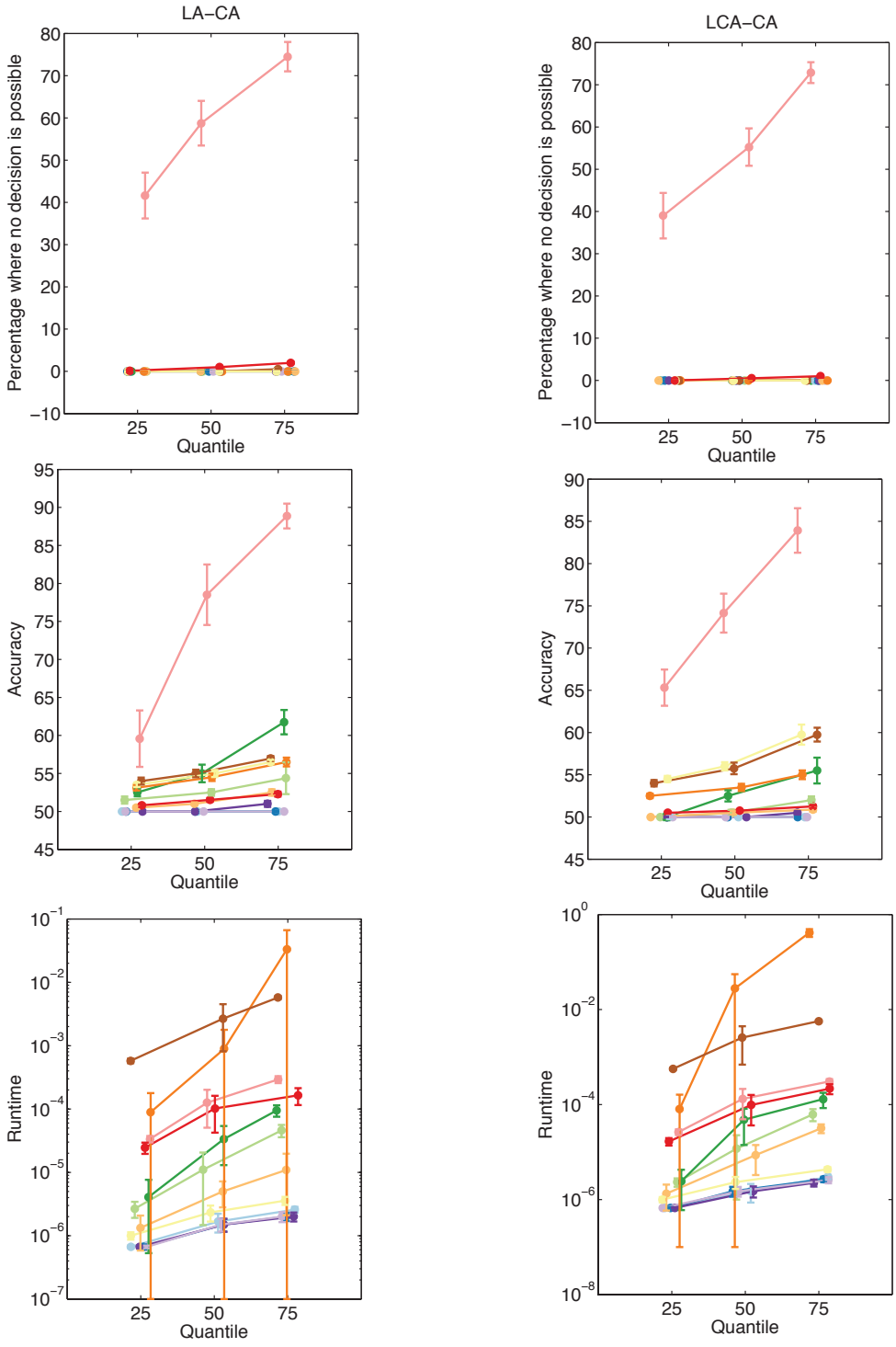
Usher, M., & McClelland, J. L. (2001). The time course of perceptual choice: The leaky, competing accumulator model. *Psychological Review*, *108*, 550-592.

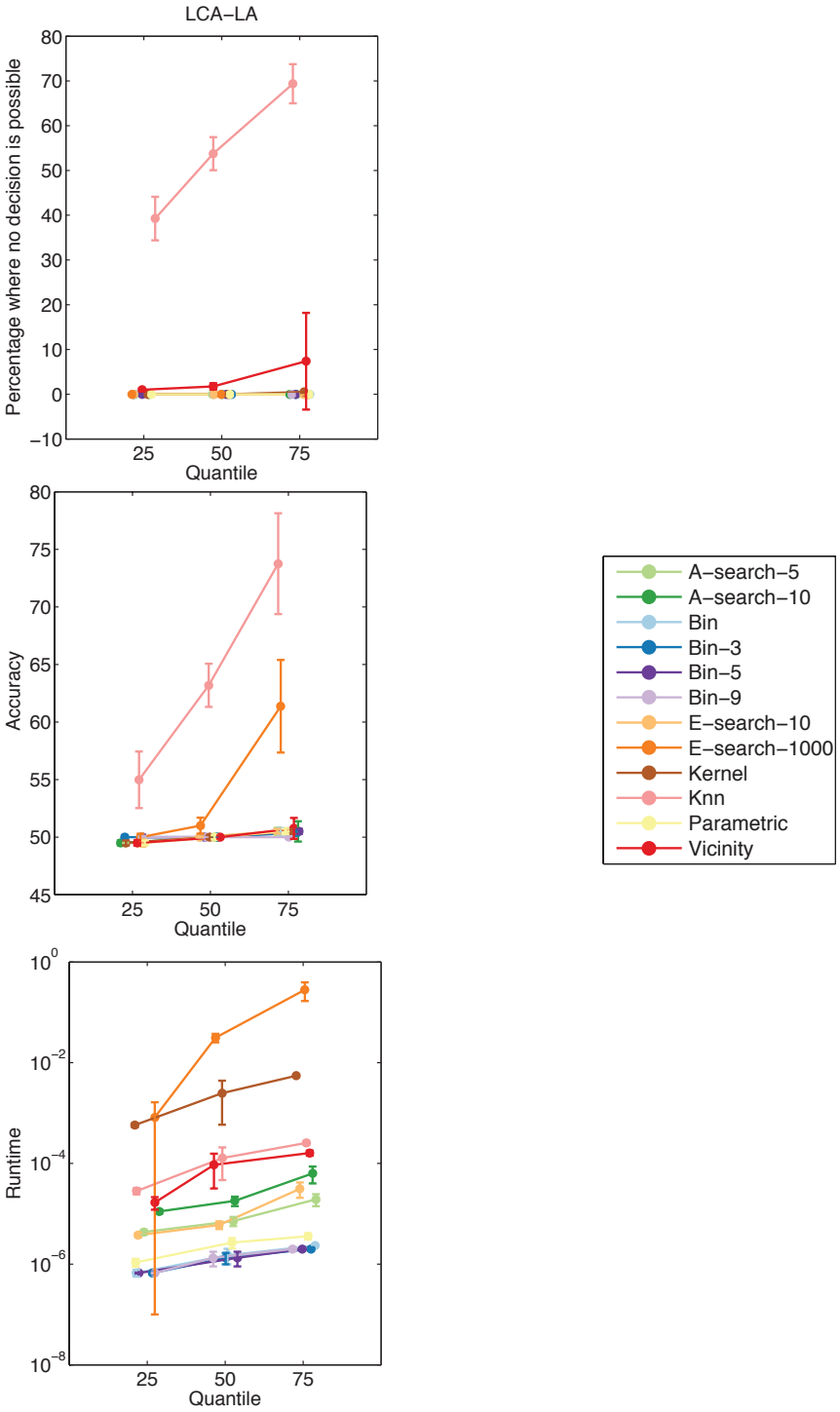
Vickers, D. (1970). Evidence for an accumulator of psychophysical discrimination. *Ergonomics*, *13*, 37 – 58.

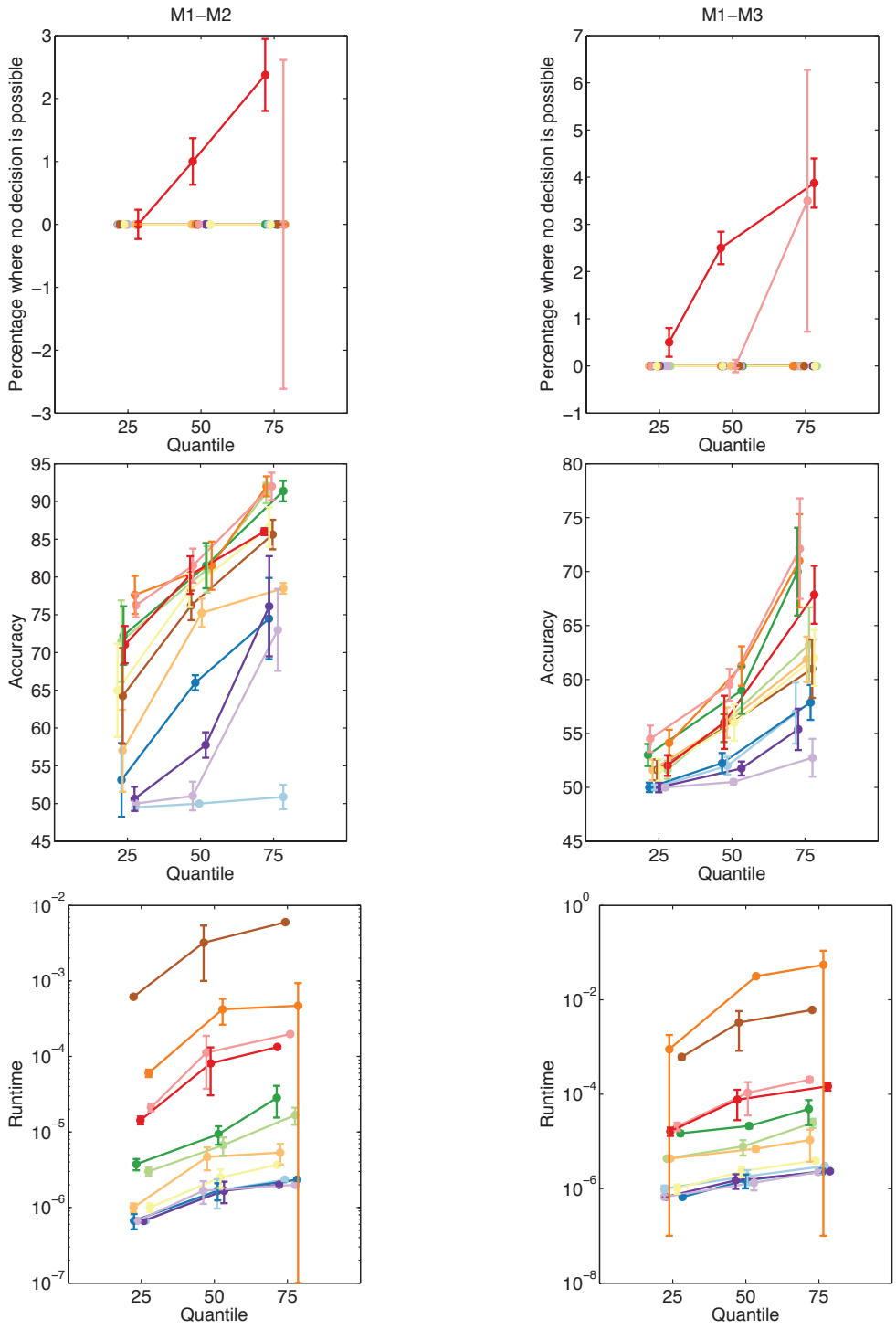
Wagenmakers, E.-J., Ratcliff, R., Gomez, P., & Iverson, G. J. (2004). Assessing model mimicry using the parametric bootstrap. *Journal of Mathematical Psychology*, *48*, 28-50.

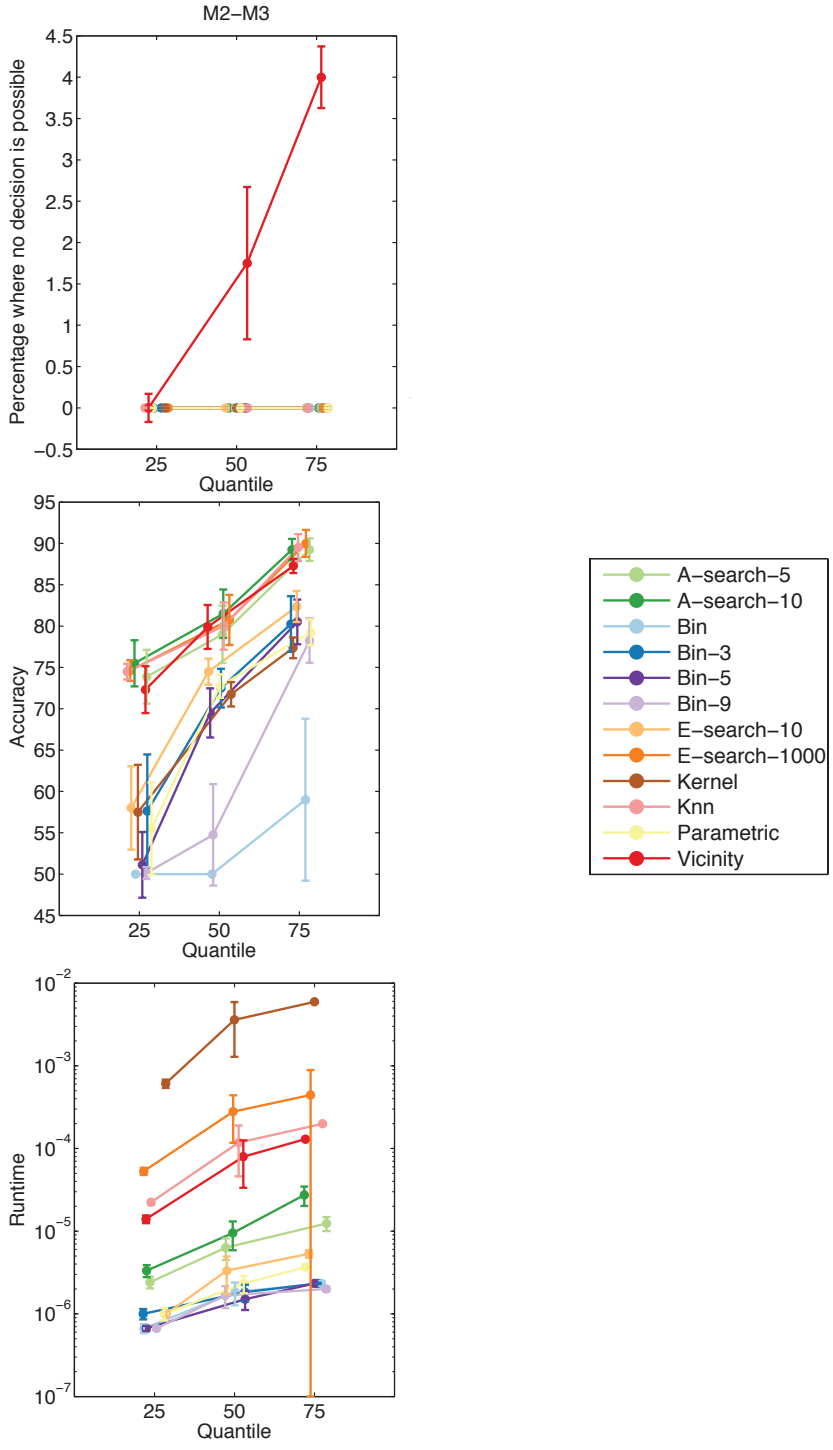
Appendix

A Results for Individual Pairs of Perceptual Choice and Memory Models

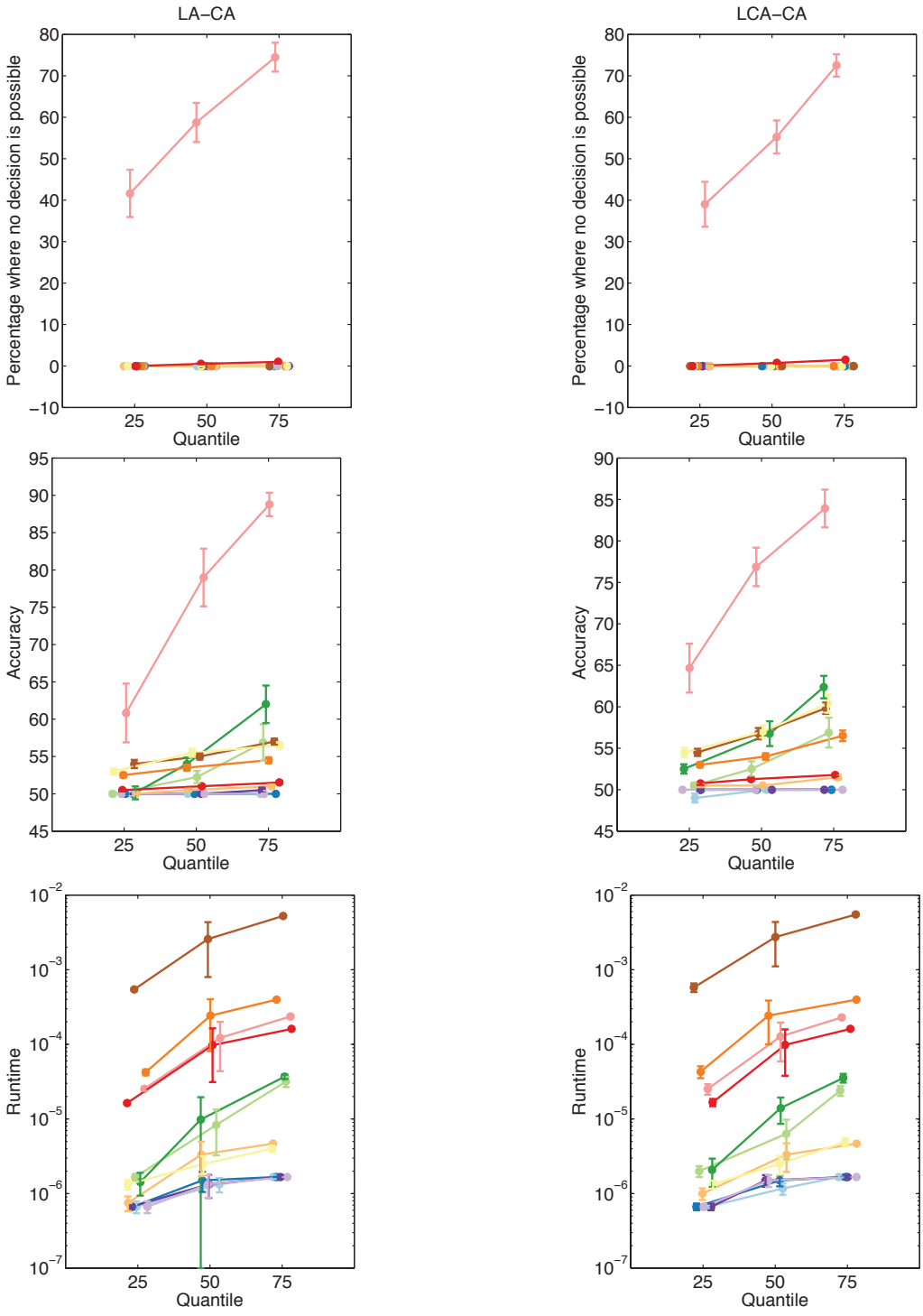


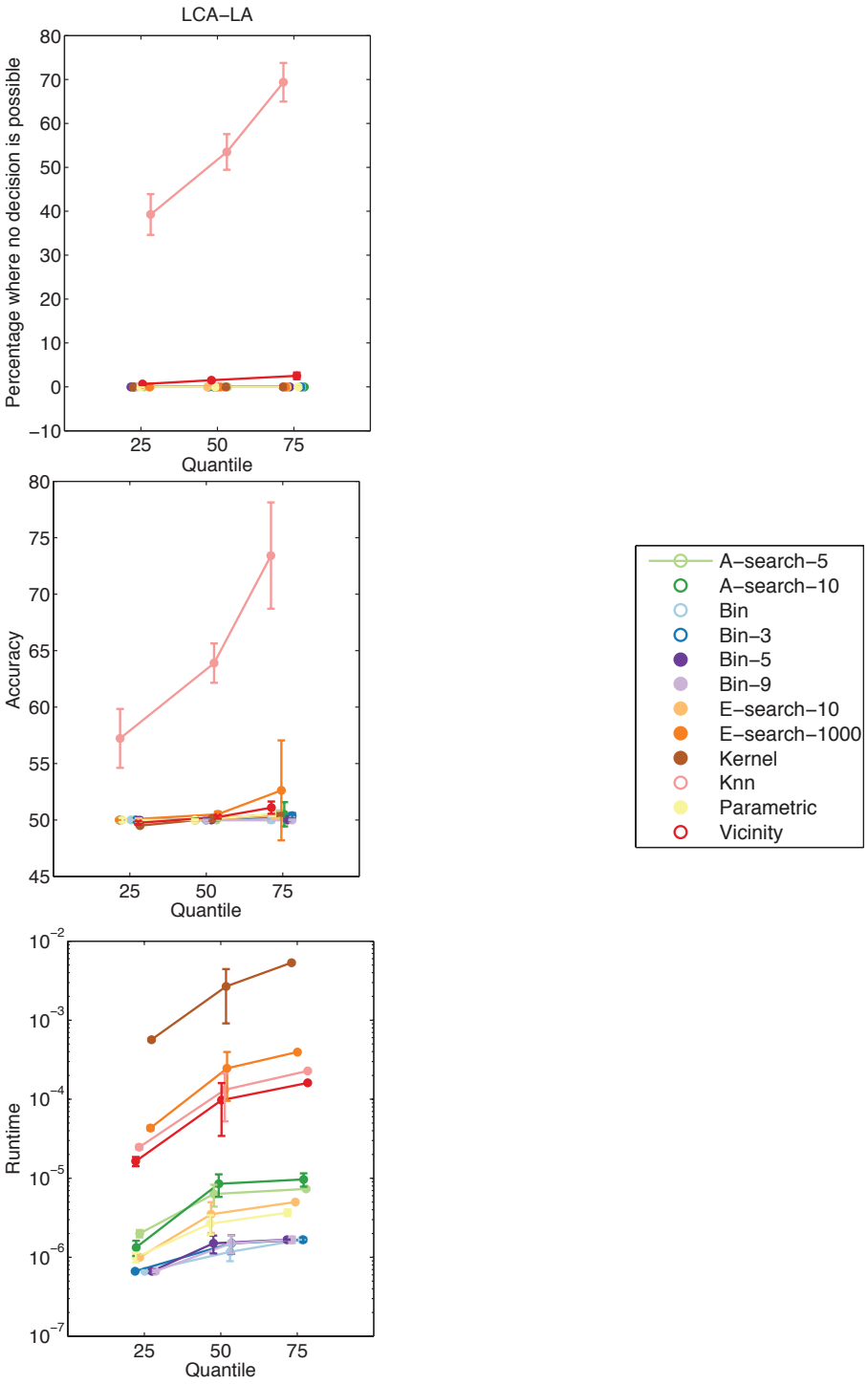

LCA–LA
Percentage where no decision is possible
80
70
60
50
40
30
20
10
0
−10
25
50
75
Quantile
Accuracy
80
75
70
65
60
55
50
45
25
50
75
Quantile
Runtime
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
25
50
75
Quantile
A–search–5
A–search–10
Bin
Bin–3
Bin–5
Bin–9
E–search–10
E–search–1000
Kernel
Knn
Parametric
Vicinity

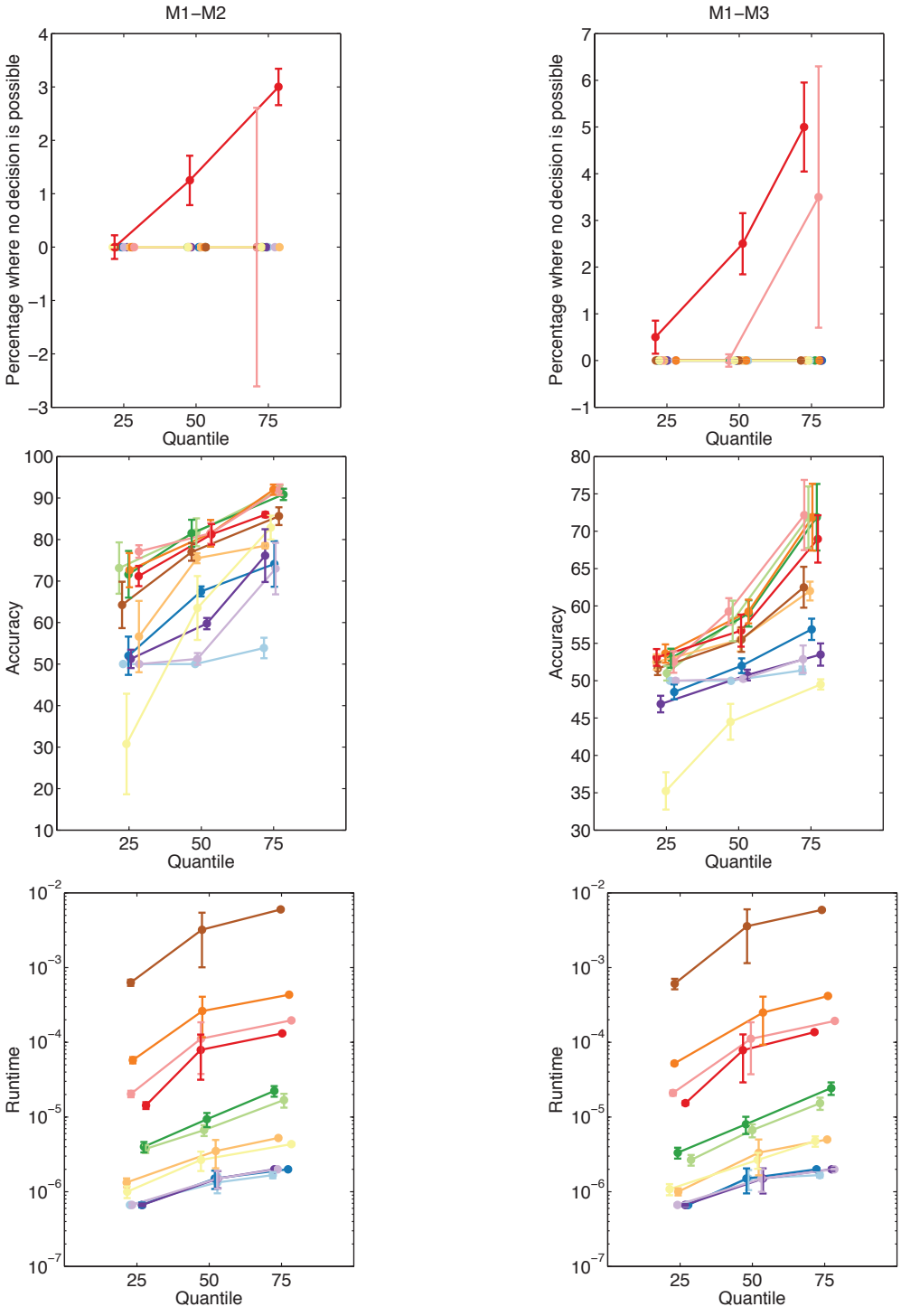

M1−M2
M1−M3
Percentage where no decision is possible
Accuracy
Runtime
Quantile
25
50
75

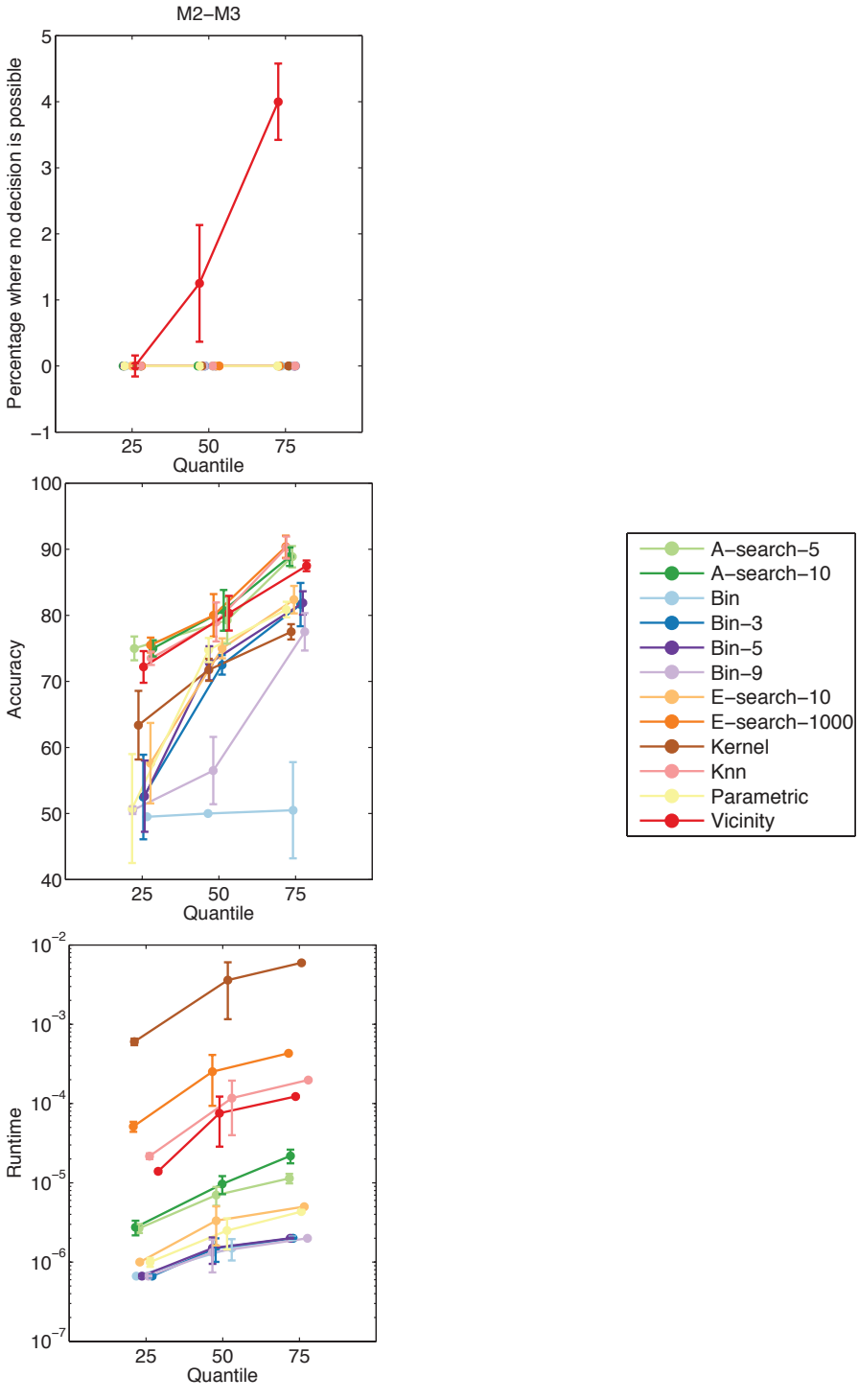

M2–M3
Percentage where no decision is possible
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
−0.5
25
50
75
Quantile
Accuracy
95
90
85
80
75
70
65
60
55
50
45
25
50
75
Quantile
Runtime
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
25
50
75
Quantile
A–search–5
A–search–10
Bin
Bin–3
Bin–5
Bin–9
E–search–10
E–search–1000
Kernel
Knn
Parametric
Vicinity

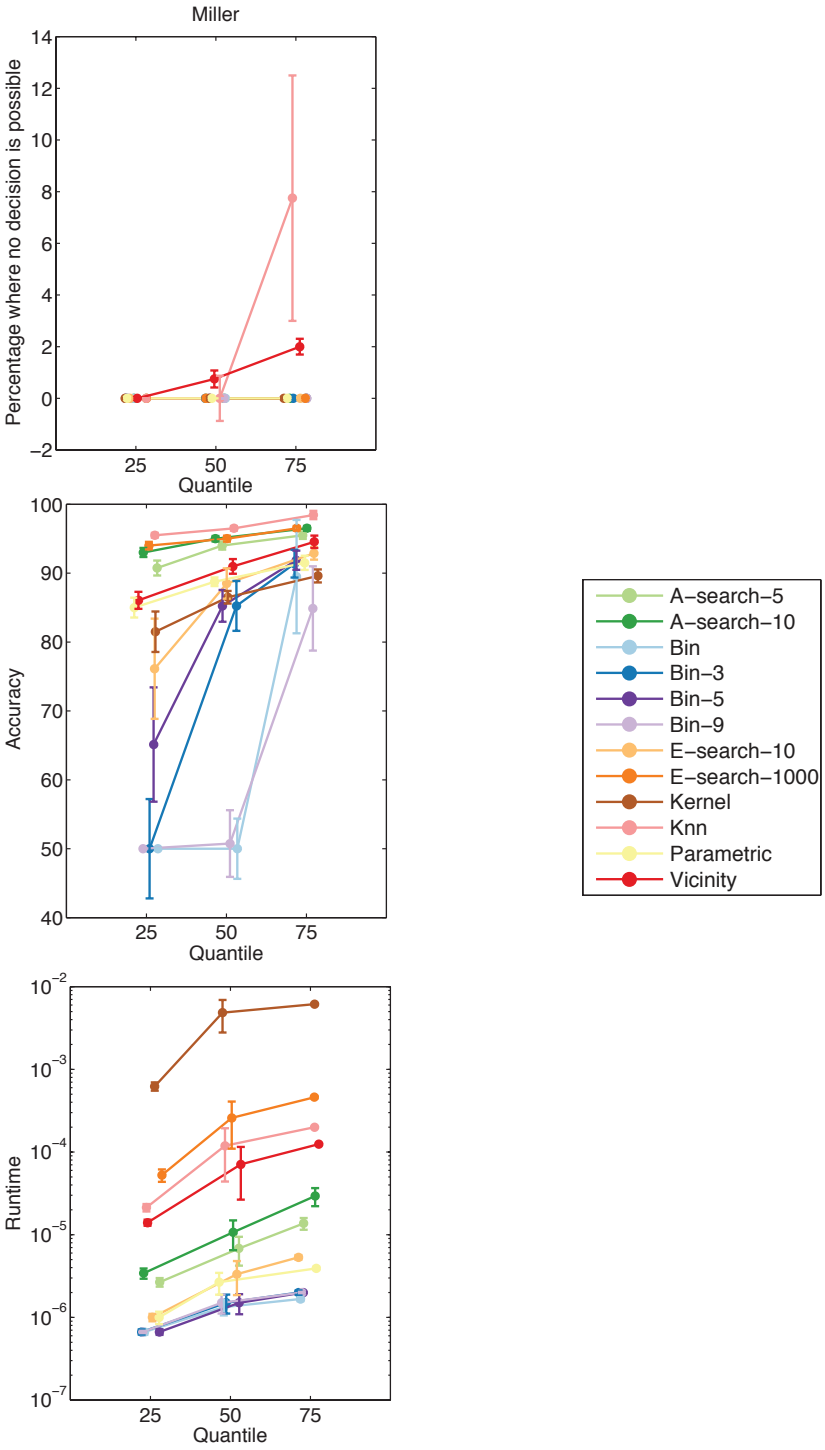
B Single Boundary Results for Individual Pairs of Perceptual Choice, Memory, and Artificial Grammar Learning Models




LCA–LA
Percentage where no decision is possible
80
70
60
50
40
30
20
10
0
–10
25
50
75
Quantile
Accuracy
80
75
70
65
60
55
50
45
25
50
75
Quantile
Runtime
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
25
50
75
Quantile
A–search–5
A–search–10
Bin
Bin–3
Bin–5
Bin–9
E–search–10
E–search–1000
Kernel
Knn
Parametric
Vicinity


M1−M2
M1−M3
Percentage where no decision is possible
Accuracy
Runtime
Quantile
25
50
75


M2–M3
Percentage where no decision is possible
Quantile
Accuracy
Quantile
Runtime
Quantile
A–search–5
A–search–10
Bin
Bin–3
Bin–5
Bin–9
E–search–10
E–search–1000
Kernel
Knn
Parametric
Vicinity


Miller
Percentage where no decision is possible
Quantile
Accuracy
Quantile
Runtime
Quantile
A−search−5
A−search−10
Bin
Bin−3
Bin−5
Bin−9
E−search−10
E−search−1000
Kernel
Knn
Parametric
Vicinity