

Supplementary Materials for

Everyday Heuristics as Bayesian Inference

Paula Parpart, Matt Jones, Bradley C. Love

correspondence to: [paula.parpart.10@ucl.ac.uk](mailto:paula.parpart.10@ucl.ac.uk), [b.love@ucl.ac.uk](mailto:b.love@ucl.ac.uk)

**This PDF file includes:**

Materials and Methods

Supplementary Text

Figs. S1 to S3

Materials and Methods

Simulation 1: Convergence of Bayesian model with heuristics (Fig. 3.A.)

This simulation demonstrates how heuristics and the Bayesian inference model converge as a function of the prior’s penalty parameter in an artificial dataset. In order to generate the Bayesian model predictions for artificial binary choices, the posterior weights (i.e. the model’s knowledge representations) were paired with either a TTB or a tallying decision rule. The artificial dataset was created as follows: We generated cue values on  cues, with a certain covariance matrix  between cues. The covariance matrix was generated such that it contained either zero (0), low (.2), medium (.5), or high (.8) values in every off-diagonal element matrix element, while the diagonal contained ones. Thus the resulting sample covariance is also either close to zero, low, medium or high, but lower than the generating covariance. For each stimulus (e.g., a stimulus could be the sedan, or the SUV, in the automobile example of the main text, Fig.1), we randomly sampled the raw cue values (e.g., in the example these are values on price, seating capacity, and gas mileage) from a multivariate Normal distribution with the given covariance matrix  and means equal to zero, and then thresholded these values at zero to discretize them into (0, 1). These binary cue values are represented by the positive and negative icons in the automobile example in the main text (Fig.1). Then we created all pairwise comparisons of the stimuli, e.g., one pairwise comparison would be comparing the sedan to the SUV on all cues. For each stimulus pair, we computed the cue-difference values. The cue differences were computed by taking the sign of the difference between the cue values of the stimulus on the left and the cue values of the stimulus on the right. Thus, the resulting values were +1 when the left stimulus was favored, -1 when the right stimulus was favored, and 0 when the cue did not distinguish between the two stimuli. Next, we created a matrix of cue differences, with one row for each stimulus pair. For this particular simulation, we sampled a set of  generating beta weights from an exponential distribution with an exponential rate of 2. However, note that although thechoice of generating beta weights influences how the penalty parameter affects model performance, the general pattern of convergence in the limit is true of any set of beta weights and corresponding dataset. We then used the cue differences, generating beta weights, and Gaussian noise to generate a criterion variable for each stimulus pair, indicating which stimulus won the comparison. If the cue-differences matrix is , the vector of beta weights , and the Gaussian noise , then the criterion variable  would be generated through matrix multiplication,





Where  is thresholded at zero into + 1 and -1 to indicate which stimulus won the competition. Thus,  is coded in the same way as the cue differences above, with -1 indicating that the right stimulus won the competition and +1 indicating the left stimulus won. All models, that is, the Bayesian learning model, the heuristics, and linear regression, were fit to this artificial dataset, and subsequently made predictions for a novel test set. The predictions on this test set were used to measure agreement by comparing the predictions of the different models. In other words, all models first trained on the artificial dataset described above and were then tested on a different dataset. The test set was constructed according to a complete sampling approach where each possible combination of  cues occurs once. For three cues that can take three values, i.e., -1,+1,0, there are 27 possible cue combinations. Each row of this cue matrix represents a stimulus pair, or a binary choice between two alternatives, like between two cars. Linear regression was run by first estimating the optimal linear regression coefficients on the training set, and then using these optimal coefficients to predict the unseen 27 test items, via matrix multiplication. The initial predictions are therefore continuous, and are discretized by taking the sign of these predictions.

The predictions by the TTB and tallying heuristics were derived by applying the respective heuristic to the cue validities as estimated from the artificial data set. Cue validities are always computed by dividing the number of times a single cue predicts the outcome  correctly by the total number of correct and incorrect predictions that the cue makes (see supplementary text 1 for details) (*1*). For each of the 27 binary stimulus pairs, the TTB heuristic makes predictions by sequentially searching through the cues in order of their validity until a cue discriminates the stimuli, and then using the discriminating cue’s value (-1,+1, or 0) as the prediction (supplementary text 2). Tallying, in contrast, simply takes the signs of the cue validities, i.e., unit weights (+1 or -1), which are then multiplied with the signs of the corresponding cue values (-1,+1, or 0) to adjust for directionalities. Tallying then adds up the resulting signs (after multiplication), and the resulting overall sign represents the prediction.

To derive the Bayesian model predictions, it was necessary to estimate the posterior distribution over the weight matrix  from the data by applying a Markov chain Monte Carlo (MCMC) algorithm to sample from the posterior. The full posterior probability distribution of our Bayesian inference model can be found in the supplementary text 3 and 4 below, detailing the MCMC algorithm in combination with our Covariance Orthogonalizing Regularization (COR). We applied a common random walk Monte Carlo method, namely the Metropolis-Hastings sampler (*2*). As we were interested in the change of the posterior weight matrix as a function of the penalty parameter in the model, we derived a different posterior estimate for each value of the penalty parameter. In each case, the mean of the posterior samples provided the estimated posterior weight matrix. This weight matrix was in turn used to make predictions with regard to the exhaustive 27 test pairs. Thus the test set is a matrix  containing  columns and 27 rows, the posterior weight matrix  is a  square matrix, and by matrix multiplication, the output is also a matrix with dimensions :



(S.2)

This output matrix  contains the posterior predictions of the Bayesian model with respect to the three output variables  in Fig. 2 of the main text. In order to convert these multivariate predictions into the model’s choices, a TTB or a tallying decision rule is applied to the output matrix as detailed in the main text. Each row of the output matrix, i.e., , represents the predictions for a stimulus pair, or a binary choice between two alternatives like two cars. All models include the guessing possibility, i.e., a choice of 0 whenever the two stimuli matched on all cues. The agreement between the Bayesian model and the TTB decision rule, or the tallying decision rule, was computed by dividing the number of equal predictions made on the test set by the total size of the test set. The agreement between the Bayesian model and traditional linear regression model was computed in the same way.

To summarize, these were the parameters of the artificial dataset as presented in Fig.3A:

* Number of cues: 
* Generating covariance level between cues: 0.8 (Average sample covariance between cues: 0.55)
* Generating beta weights: [0.43, 0.22, 0.26] (Sample cue validities: [0.5, 0.36, 0.17])
* Number of pairwise comparisons to train on: 20
* Number of pairwise comparisons to test on: 27
* Penalty parameter values tested:  = [0.01, 0.1, 0.5, 1, 2, 3, 4, 5, 6, 10, 30, 50, 80, 100]

Simulation 2: Generalization performance of the Bayesian model (Fig. 3.B.)

The goal of this simulation is to explore the performance of the Bayesian inference model in cross-validation under particular environmental conditions. In particular, this simulation tests the dependence of the Bayesian models’ generalization performance on the number training pairs, i.e. the number of pairwise comparisons. The expectation was that with more data, more complex models (i.e., with smaller penalty parameters) that freely estimate co-variance should fare better, whereas the simpler models (i.e., with higher penalty parameters) should have an advantage with less data.

We generated a large dataset of pairwise comparisons with  using the same method as in simulation 1. The data were randomly split into a training set and a test set by means of repeated random sub-sampling cross-validation. The size of the training set depended on the condition and was varied as indicated below – for example, a small training set contained only 20 pairwise comparisons, and a large training set contained 100 pairwise comparisons. The size of the test set was complementary to the size of the training set (i.e., the test set comprised all remaining pairs). The Bayesian learning model estimated the weight matrix from the training data and was then cross-validated by predicting the test set. As detailed in simulation 1, to estimate the posterior weight matrix, we relied on a MCMC algorithm (supplementary texts 3 and 4). Since the Bayesian prior’s strength is regulated by a penalty parameter, we estimated a posterior distribution for each value of that parameter. At the prediction stage, for each penalty value, the estimated posterior weight matrix was used to make predictions with respect to the test set, in the same manner as in Eq. 2. To assess the model’s predictive accuracy, the predictions were then compared to the actual criterion variable of the test set, which contains the binary outcomes (generated as in simulation 1).

To summarize, these were the parameters of the artificial dataset as presented in Fig.3B:

* Number of cues: *m* = 5
* Generating covariance level between cues: 0.5 (Average sample covariance between cues: 0.23)
* Generating betas weights: [1, 0.8, 0.8, 0.7, 0.6] (Sample cue validities: [0.28, 0.39, 0.45, 0.30, 0.30])
* Total number of pairwise comparisons: .
* Number of pairwise comparisons to train on: 20 (small) or 100 (large)
* Number of pairwise comparisons to test on: *N* - 20, *N* - 100
* Penalty parameter values tested:  = [0, 0.1, 1, 5, 6, 7, 8, 9, 10, 15, 20, 40, 50, 70, 100]

Supplementary Text

1 Cue validities

Cue validities are defined for binary decision tasks, wherein two stimuli are compared on the dimensions of several cues and the inference is made about which stimulus has the higher criterion value. The outcome variable encodes which cue actually won the competition, and could be coded for example as -1 and +1 as in Figure 1 (main text). Cue validity *v* is defined as the proportion of correct predictions that a single cue makes (*1*)**:**



*R* = number of correct predictions,

*W* = number of wrong predictions.

and consequently,.

A cue with validity less than 0.5 has been called invalid (i.e., equivalent to a regression coefficient beta below zero) in the literature, and in that case the cue’s values are usually inverted, e.g., if it was +1 before it would be coded as -1 (*1*). Cue validities are a linear transformation of single-predictor regression coefficients when outcomes are binary (*1*) - they are centered at 0.50 instead of 0 for regression coefficients. For our purpose, we defined cue validities as above, but we took the difference between each cue validity and 0.50, i.e.**, ,** in order for cue validities to have the same signs as the corresponding single-predictor regression coefficients. Note that cue validities are computed in isolation, i.e., not taking into account any covariance among cues. They reflect how good each cue is at making correct inferences about the criterion, independently of the other cues. Just like single-predictor regression coefficients, cue validities can display different rank orders and signs than multiple regression coefficients from the same dataset (see Figure S1). This is due to covariance among cues, which affects the regression coefficients in multiple linear regression, but not cue validities.

2 Heuristic Mechanisms

2.1 Tallying Heuristic

The tallying heuristic is defined for binary decision tasks, wherein two alternatives are compared on several cues and the inference is made about which alternative has the higher criterion value. Tallying works by ignoring all weights and simply counting up the number of cues favoring one alternative in comparison to the other (*3*). It decides for the alternative with more favoring cues. For the case where the number of cues favoring either alternative is equal, it makes a guess. For example, in deciding between two cars, the number of pieces of positive evidence for car A and for car B would be counted up, and the decision would be to buy the car with the greater number of pieces of positive evidence. Tallying was originally called Dawe´s rule, which is a linear model that uses only unit weights (+1 or -1) (*3*). In Dawes’ original definition he assumed that people already know the directionalities of the cue weights. However, since then others have operationalized the heuristic assuming people do not know the cue directionalities in advance and still need to learn them from the environment (e.g., via estimation from the training sample) (*4*). This is what we have done in our simulations too. It has been shown that a tallying strategy does better in an environment where the weighting structure of cues is more homogenous, which is often called a *compensatory* environment (*1*).

2.2 Take-The-Best Heuristic

Take-The-Best is defined for binary decision tasks, wherein alternatives are compared on several cues, and the inference is made about which alternative has the higher criterion value. The TTB heuristic assumes that people first order cues by their cue validities. Thus, like other fast-and-frugal heuristics, take-the-best is often defined according to a search rule, a stopping rule and a decision rule (*5*), as follows. (1) Search rule: Search through cues in order of their validity. (2) Stopping rule: Stop upon finding the first cue that discriminates the alternatives. (3) Decision rule: Infer that the alternative with the higher value on that cue has the higher criterion value. Take-the-Best is a *non-compensatory* strategy, because the highest ranked, i.e., most valid, cue  will outweigh any combination of the subsequent cues . This means that once the most valid discriminating cue has been found, it is used to make the inference and no further cues are considered. Hence, TTB is a strategy that ignores a large part of the information. Like tallying, it does best in its corresponding environment, i.e. in an environment that also has a *non-compensatory* weighting structure (*1*).

3 Markov Chain Monte Carlo (MCMC)

We used a Metropolis-Hastings sampler to generate Markov chainsin order to estimate the Bayesian posterior weight matrix. MCMC algorithms are a class of methods for generating samples from complex probability distributions by constructing Markov chains that converge to those distributions over time (*6*). For drawing a sample from a certain probability distribution, the Markov chain is defined such that the stationary distribution of that chain is the desired distribution, and a sequence of states is then sampled from that chain. If the sequence is long enough, the states of the chain can be treated similarly to samples from the equilibrium distribution. The Metropolis algorithm (*2*) is one of the most popular methods for constructing such a Markov chain. The sequence of states is initialized with an arbitrary value, . The next value in the sequence is generated via a two step process. First, a candidate for the next value, , is chosen by sampling from an arbitrary proposal distribution conditioned on , that is specified by the designer of the algorithm. We used a multivariate Gaussian proposal density*.* Second, a decision is made as to whether that proposed value will be accepted, using a valid acceptance function. For this purpose, an *acceptance ratio* α is computed,



If  , then the candidate is more likely than *x,* and we automatically accept the candidate by setting . Otherwise, we accept the candidate with probability α, and if the candidate is rejected, we set  instead.

The target distribution used here is in accordance with the posterior probability distribution over the weight matrix that is derived in the supplementary text 4 on Covariance Orthogonalizing Regularization (COR):





And the log unnormalized posterior distribution of the target distribution is





4 Mathematical Derivations

1. **Penalized Regression as Bayesian Inference**

We treat penalized regression as a Bayesian inference problem. Thus, first consider a problem of inferring a vector *w* from a set of observations  under the assumptions

1. the inputs  are independent of *w*

2. the inputs  and  are independent for 

3. the outputs  and, for , are conditionally independent given , and *w*.

Define the conditional distribution of  by



For example, linear regression is defined by 



Given a prior on *w*, the posterior is given by





If the goal is to find the value of *w* that maximizes the posterior, then the problem can be formulated as



The formulation in Eq. 9 corresponds to a general penalized regression problem. Different choices of  correspond to different regression models, and different choices of  correspond to different penalty terms.

1. **L2 Penalty term**

**Gaussian prior = L2 regularization**

Let the prior on *w* be Gaussian such that its components are independent and identically distributed:



The optimization problem from Eq. 9 can thus be written as





Thus we have an L2 penalty term .

1. **Ridge Regression**

**Linear Regression with L2 Regularization**

Consider the case when the prior is chosen to be Gaussian as in Eq. 10, and the likelihood is as in OLS regression,



or, in different notation,



with 

with  considered a row vector and *w* a column vector. Then the optimization problem is



which is equivalent to **ridge regression**. In particular, ridge regression has the following form:





Setting the penalty parameter  to makes Eq. 13 and 14 equivalent.

The term  is the loss term and is a measure of the goodness of fit to the data, whereas the term  is the penalty term. As , will approach 0. However, when ,  returns the ordinary least-squares (OLS) regression estimates.

1. **Covariance Orthogonalizing Regularization (COR)**

**Multivariate Regression with COR Regularization**

In a multivariate regression problem, there are several input variables and output variables, and hence the goal is to estimate a weight matrix. Therefore, we switch to matrix notation. For example, in a scenario with  input variables, or cues, and  output variables, a  weight matrix would be estimated,

 ,

where each column represents the regression weights from regressing one component of the output  onto all  cues. Without any penalization, these weights are equal to OLS regression coefficients. The diagonal weights  in the weights matrix  reflect the weights from regressing each cue directly onto , and if no other cues are present, these weights are equivalent to single-predictor regression coefficients (as represented by solid arrows in Fig. 2 of the main text). Single-predictor regression coefficients have the same rank order and directionality as heuristic cue validities, in fact, the cue validities are a positive rescaling of the single regression coefficients with binary cues and a binary criterion variable (*1*). When the off-diagonal weights are non-zero, it means that the regression model contains more than just one cue, and naturally, covariance among cues is estimated as part of the regression coefficient estimation. For the present application, the outcome variables in the multivariate regression problem were derived by replicating the outcome variable  as often as there are input variables , which leads to exactly equivalent output variables that make up an output matrix  (see Fig. 2 in main text).

The multivariate regression model  follows a multivariate Gaussian distribution. The logarithm of a multivariate Gaussian has the following form:



where *C* is the covariance matrix of the stochastic component of the model (i.e., ε, which is now a random vector), and *c* is a constant that is independent of *W*, *X*, and *Y*.

In our COR regularization, the prior over the weight matrix,, and its corresponding penalty term are defined in analogy to ridge regression, where the magnitudes of the weights is regulated by a penalty parameter. In this case, the penalty is applied to the off-diagonal elements of the weight matrix only (Eq. 15), and not to the diagonal elements. Note that when the off-diagonal weights  are zero, the remaining direct weights  do not reflect any covariance information. Hence, penalizing these off-diagonal weights results in regularization of the covariance sensitivity in the model. In the limit, when , the off-diagonal weights are shrunk to zero, while the weights in the matrix diagonal  stay unpenalized.

The penalization is expressed in the logarithm of the prior over the weight matrix ,



where  is the penalty parameter again, and the L2 matrix norm is applied to the off-diagonal elements of the weight matrix  this time. Applying a (squared) Euclidean norm to a matrix (treating it as a vector) is equivalent to applying the (squared) Frobenius norm, i.e., , to the weight matrix.

Combining the log likelihood (Eq. 16) and the log prior (Eq. 17) results in the posterior,





This represents the log (unnormalized) posterior distribution which was relied on for sampling from the posterior probability distribution using an MCMC sampling algorithm (as explained in supplementary text 3).

The posterior probability distribution can also be formulated as the product of prior and likelihood,





To summarize the Bayesian posterior distribution, we evaluated the mean of the posterior, which is expressed by the expectation of the posterior probability distribution with respect to the weight matrix **.** In order to derive this mean estimate we used a MCMC simulation method, namely the Metropolis-Hastings sampler, as detailed in the supplementary text 3 above.

**1 2 3 4**

**0.00 1.00 0.25 0.71**

**1 2 3 4**

**-0.29 1.16 -0.11 0.08**

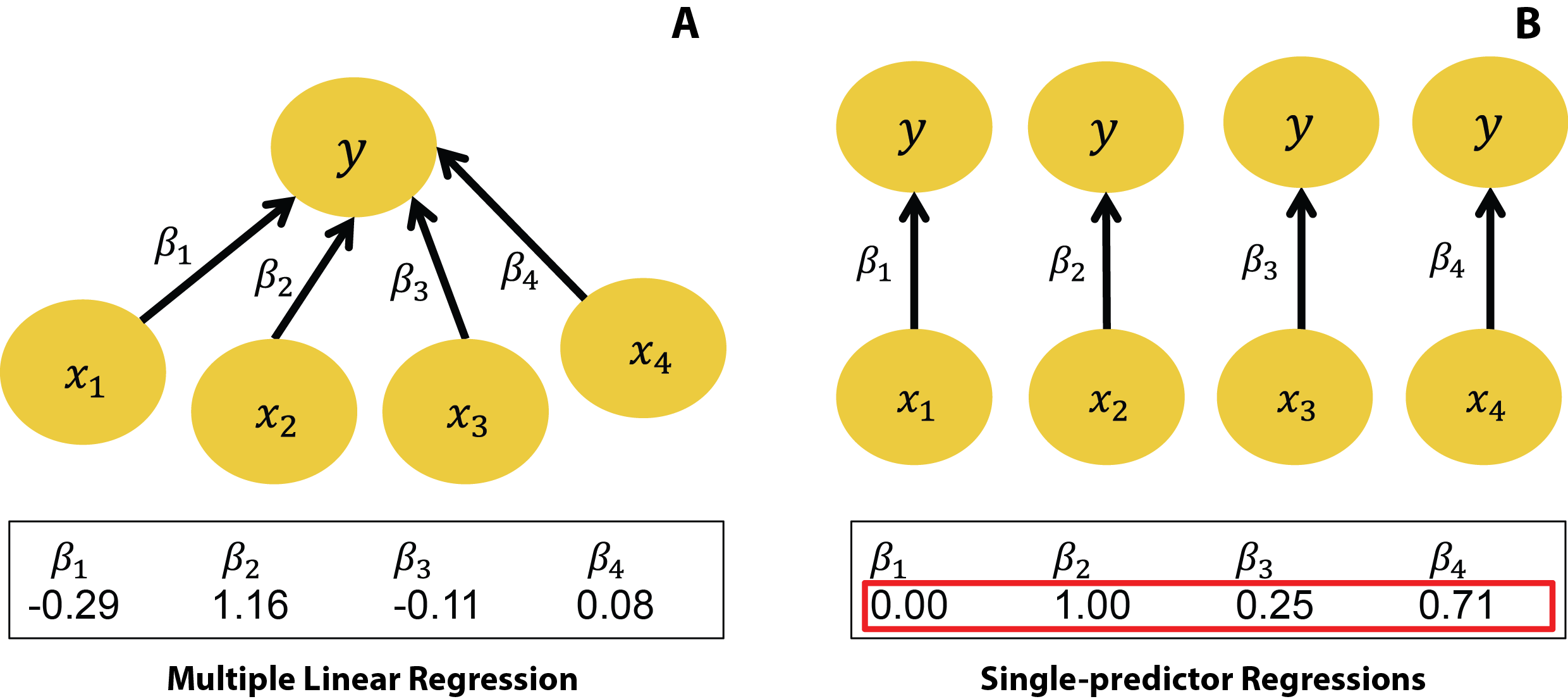
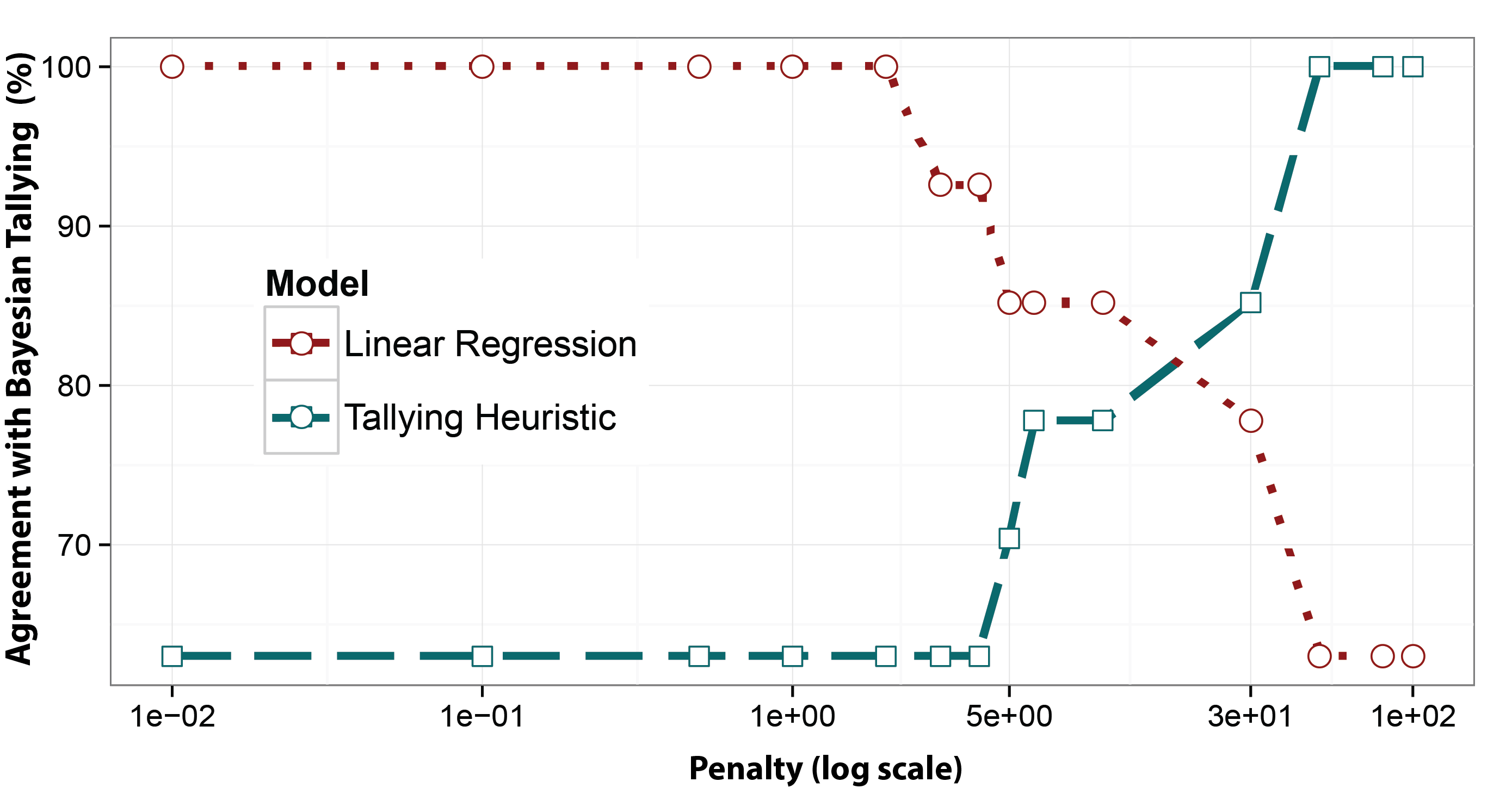
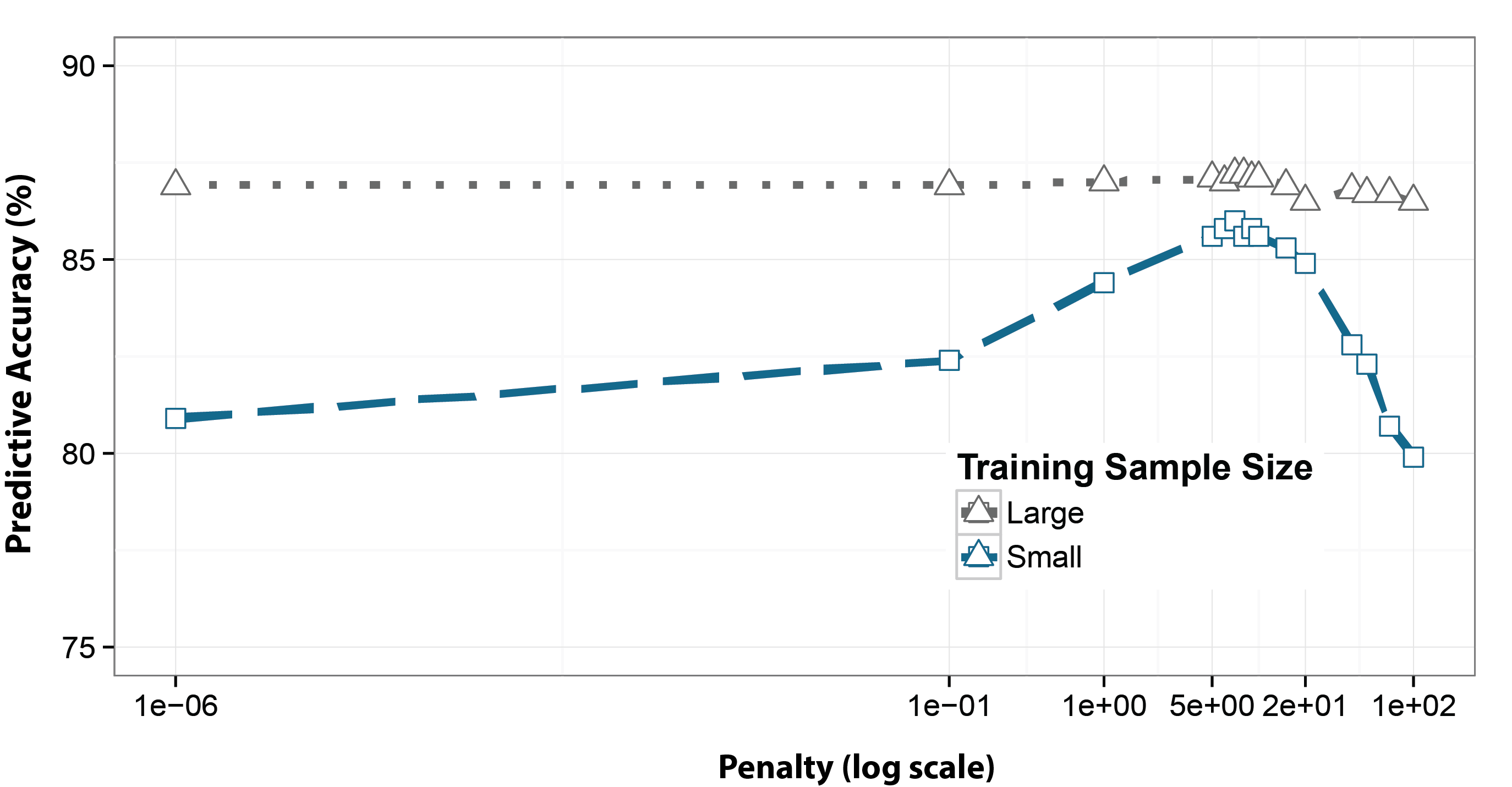


Fig. S1

**(A)** An example of a multiple linear regression problem with four predictor variables X1 to X4, and one outcome variable Y. The estimated parameters, i.e. regression coefficients, are shown below and are naturally taking into account the co-variance among the predictor variables. **(B)** Four independent single-predictor regression problems where each predictor X1 to X4 predicts the outcome Y on its own, i.e., independent of the other predictors. In a single-predictor regression, the regression coefficients have different magnitudes and signs from the coefficients in multiple regression, due to the single-predictor regression analysis’s ignoring co-variance among predictors. For example, the first coefficient X1 has decreased in magnitude to zero, indicating that the effect was previously carried by the variance shared with the other predictor variables (i.e., covariance). The cue validities that the tallying and take-the-best heuristics use are like single-predictor regression coefficients, as they also estimate the impact of each predictor in isolation.

**Fig. S2**

Agreement between the Bayesian model’s Tallying decision rule and the Tallying heuristic, as well as linear regression, as a function of penalty parameter. The vertical axis reflects the percentage agreement during prediction in an artificial dataset. The dataset was exactly the same as shown in Fig. 3A (Bayesian TTB decision rule) of the main text, and its underlying parameters are detailed in the materials & methods section above (Simulation 1).

Fig. S3

Generalization performance of the Bayesian Tallying decision rule in cross-validation: The Tallying decision rule performs better with smaller penalties when training samples are large, and performs best for an intermediate penalty when training samples are small. Parameters of the artificial dataset were equivalent to those used in Fig. 3B (Bayesian TTB decision rule) of the main text, and are detailed above in the materials & methods section (Simulation 2).

1. L. Martignon, U. Hoffrage, in *Simple heuristics that make us smart*. (Oxford University Press, New York, 1999), pp. 119-140.

2. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, Equation of state calculations by fast computing machines. *The journal of chemical physics* **21**, 1087-1092 (1953).

3. R. M. Dawes, The robust beauty of improper linear models in decision making. *American psychologist* **34**, 571 (1979).

4. J. Czerlinski, G. Gigerenzer, D. G. Goldstein, in *Simple heuristics that make us smart,* G. Gigerenzer, P. M. Todd, A. R. Group, Eds. (Oxford University Press, New York, 1999), pp. 97–118.

5. G. Gigerenzer, D. G. Goldstein, Reasoning the fast and frugal way: models of bounded rationality. *Psychological review* **103**, 650-669 (1996)

6. W. R. Gilks, S. Richardson, D. J. Spiegelhalter, in *Markov chain Monte Carlo in practice*. (Springer, 1996), pp. 1-19.