webppl-oed: A practical optimal experiment design system

Long Ouyang*, Michael Henry Tessler*, Daniel Ly, Noah D. Goodman
Departments of Psychology and Computer Science
Stanford University
Stanford, CA 94305
{louyang, mhtessler, ngoodman}@stanford.edu

Abstract

An essential part of cognitive science is designing experiments that distinguish competing models. This requires patience and ingenuity—there is often a large space of possible experiments one could run but only a small subset that might yield informative results. We need not comb this space by hand: If we use formal models and explicitly declare the space of experiments, we can automate the search for good experiments, looking for those with high *expected information gain*. Here, we present an automated system for experiment design called webppl-oed. In our system, users simply declare their models and experiment space; in return, they receive a list of experiments ranked by their expected information gain. We demonstrate our system in two case studies, where we use it to design experiments in studies of sequence prediction and categorization. We find strong empirical validation that our automatically designed experiments were indeed optimal.

Introduction

Cognitive scientists often design experiments to test competing computational models. Good experiments are ones where the models make different predictions, but there is typically a large space of possible experiments one could run (e.g., there could many different possible stimulus sets to present). Rather than systematically search the experiment space, scientists often rely on intuition to design experiments where models sufficiently diverge. This intuition may be biased in a number of ways, such as towards experiments that show qualitative differences between models even when more informative quantitative differences may exist.

In principle, there is a better way—if we formally declare the space of models and space of experiments, optimal experiment design (OED) allows us to automate the search for good experiments (i.e., ones that strongly update our beliefs about a scientific question). However, while the mathematical foundations of OED are fairly straightforward (Lindley, 1956), it has not enjoyed widespread use in practice. Some OED systems are too specialized for general use; others are more general but require too much statistical and computational know-how to be widely adopted (e.g., users must supply their own objective function and derive a solution algorithm for it). In this work, we describe an automated system that is both general and practical—the user writes the competing models and space of possible experiments in a common language; a set of potentially informative experiments is then computed with no further input from the user.

We first describe our framework in general terms and then apply it in two case studies. First, we consider the problem of distinguishing three toy models of human sequence prediction. Second, we go beyond toy models and analyze a classic paper (Medin & Schaffer, 1978) on human category learning that compared two models using a hand-designed experiment. Our OED system discovers experiments that are several times more effective (in an information-theoretic sense) than the original. Our work opens a number of rich areas for future development, which we explore in the discussion.

* Authors contributed equally to this work.

Experiment design framework

Imagine that we are studying how people predict elements in a sequence (e.g., flips of a possibly-trick coin). We want to compare two cognitive models of people's behavior: m_{fair}, where people believe the coin is unbiased (i.e., H and T are equally likely), and m_{bias}, where people believe the coin has a bias that is unknown to them. *A priori*, one model is not preferred over the other; in Bayesian terms, we have a uniform prior on the models. We wish to update our beliefs about these models through an experiment where we show people 4 flips of the same coin and ask them to predict what will happen on the next flip. There are 16 possible *experiments* (i.e., sequences of H and T for the 4 flips);[1] as each participant responds by predicting either H or T, there are 2^n possible *outcomes* for n participants. The models predict how people will respond in an experiment (i.e., after seeing some particular sequence of flips). Formally, a model defines a probability distribution on $\{\text{H}, \text{T}\}^n$ conditional on the experiment x. For convenience, we write our models in terms of what a single person would do and assume that all people respond according to the same model, i.e., participant responses are i.i.d.[2]

Should we run experiment HHTT? m_{fair} always predicts H and T with equal probability; for this experiment, m_{bias} learns that the bias favors H and T equally, and thus also makes the same prediction. Regardless of the observed outcome (person predicts H or T), the data cannot update our beliefs about the models, so this is a poor experiment. By contrast, the experiment HHHH would be much more informative. Under m_{fair}, $p(\text{H}) = \frac{1}{2}$ but under m_{bias}, $p(\text{H}) = \frac{5}{6}$ (this is the updated probability of heads after estimating the coin bias). In this case, *either* experimental response would be informative. If the participant predicted heads, this would favor m_{bias} and if she predicted

[1] Our notion of "experiment" is quite general, including traditional components like stimulus properties (e.g., coin sequence) as well as other components like dependent measure and sample size.

[2] We use this simple linking function throughout this paper but our approach handles arbitrary linking functions (e.g., hierarchical models with participant-wise parameters).

tails, this would favor m_{fair}. HHHH is a better experiment than HHTT for disambiguating the models. The goal of OED is to automate this kind of reasoning.

Now, we provide formal details. We wish to compare a set of models M in terms of how well they account for empirical data. A model m is a conditional distribution $P_m(Y \mid X)$ representing the likelihood of empirical results y for different possible experiments x. We adopt a Bayesian model comparison approach—we start with a prior on models $P(M)$ and seek an experiment whose result will maximally update this prior. *A priori*, we do not know what will happen if we run experiment x. If we were to actually run x and obtain result y, then we could measure *actual information gain* (AIG) by:

$$\text{AIG}(x) = D_{\text{KL}}(P(M \mid Y = y, X = x) \parallel P(M)) \qquad (1)$$

where D_{KL} is KL-divergence. We can compute the *expected information gain* (EIG) of x by imagining hypothetical different experimental results and combining them—that is, we can marginalize over y:

$$\text{EIG}(x) = \mathbb{E}_{p(y;x)} D_{\text{KL}}(P(M \mid X = x, Y = y) \parallel P(M)) \qquad (2)$$

where $p(y;x)$ is the probability of observing y for x. If we believe that M contains the true model of the data, then a suitable choice for $p(y;x)$ is the predictive distribution implied by the models: $p(y;x) = \mathbb{E}_{p(m)} p_m(y \mid x)$. If, however, we believe that M does not contain the true model, then an uninformative distribution $p(y;x) \propto 1$ may be more appropriate.

Note that we commit to a Bayesian approach only for model comparison; the models themselves need not be Bayesian nor even probabilistic. If the models define a probability distribution specifying predictions for different experiments, they can be be used without further assumptions. Models that make deterministic predictions can be made into such a probability distribution by having predictions serve as the mean of subjects' responses with actual responses being normally-distributed around this value; indeed, this implicit assumption underlies standard data analysis used for such models. Finally, OED does not need to do exact Bayesian computation to be useful—approximate OED can still find experiments that outperform those designed by hand.

Writing models as probabilistic programs

Models are probability distributions. As such, we have the user express their models in a programming language where probability distributions and operations on them are first-class objects. In particular, we use WebPPL (webppl.org), a small but feature-rich probabilistic programming language embedded in Javascript (Goodman & Stuhlmüller, 2014). WebPPL supports sampling from primitive probability distributions and combining these samples in various ways, e.g., adding Gaussian noise to a Binomial random variable:

```
var g = function() {
  var x = sample(Binomial({n: 4, p: 0.5}))
  var y = sample(Gaussian({mu: 0, sigma: 1}))
  x + y // function returns its last expression
}
Infer(g)
```

The function g defines a sampling procedure for our compound distribution. This *implicitly* represents a probability distribution; to reify this into an actual distribution, we must perform inference via Infer(g, options). This separation between *what* we wish to compute from *how* we try to compute it is useful when writing larger, more complex models. Note that in the above snippet, and throughout, we omit the options object, which specifies which inference algorithm to use.[3]

WebPPL also supports expressing *conditional* probability distributions. For instance, in the model above, we might ask what values of x and y could lead their sum to be greater than or equal to 2:

```
var g = function() {
  var x = sample(Binomial({n: 4, p: 0.5}))
  var y = sample(Gaussian({mu: 0, sigma: 1}))
  condition(x + y >= 2)
  [x, y]
}
Infer(g)
```

Here, condition rejects any states where $x + y < 2$.

Given a set of competing models written as WebPPL functions, a space of possible experiments (inputs to the models), and expectations about the results for different experiments (i.e., $p(y;x)$, again written in WebPPL), our system webppl-oed searches for experiments that have high EIG as defined in Eq. 2. In abstract terms, webppl-oed calculates $\text{EIG}(x)$ by sampling imagined experiment results from $p(y;x)$. For each sample y, it performs inference to obtain a posterior distribution on models and then measures the KL divergence of this posterior from the prior. The average of these KL divergences is an estimate of EIG. The main bottleneck in this process is posterior inference, which takes a good deal of expertise to implement correctly and is also computationally challenging. webppl-oed insulates end users from these technical concerns, allowing them to concentrate on scientific questions rather than engineering details. The software is available online at github.com/mhtess/webppl-oed. We next demonstrate it by using it to distinguish toy models of sequence prediction.

Case study 1: Sequence prediction

Human judgments about sequences are surprisingly systematic and nonuniform across equally likely outcomes – for example, we might strongly believe the next coin flip in the sequence HHTTHHTT will be H, whereas the sequence THHTHTHT is less suggestive of a particular next outcome. Several hypotheses have been articulated about what underlies human intuitions about such sequences (Falk, 1981; Goodfellow, 1938; Griffiths & Tenenbaum, 2004). Here, we consider three simple models of people's beliefs: (a) *Fair coin*: people assume the coin is fair, (b) *Bias coin*: people believe the coin has some unknown bias that they can estimate from data (i.e., learning the probability of an H outcome), (c) *Markov coin*: people believe the coin has some probability of transitioning between spans of H and T outcomes, also learnable from observations.

[3] WebPPL currently provides these inference algorithms: MCMC (MH, HMC), SMC, enumeration, and variational inference.

As in our earlier example, we consider an experimental setup where participants see four flips of the same coin and must predict the next flip.

Formalization

The model space M is $\{m_{\text{fair}}, m_{\text{bias}}, m_{\text{markov}}\}$. For now, we assume that the experiment will collect data from just a single participant, so the experiment space X is the Cartesian product $\{1\} \times \{\texttt{H},\texttt{T}\}^4$ representing the fixed sample size of 1 and sequence space. Finally, Y is the response set $\{\texttt{H},\texttt{T}\}^1$.

Under m_{fair}, people assume that the coin always has an equal probability of coming up heads or tails:

```
var fairCoin = function(seq) {
  Infer(function(){ flip(0.5) })
}
```

Here, `flip(0.5)` is shorthand for `sample(Bernoulli(p:0.5))`. Note the type signature of this model—it takes as input an experiment (`seq`) and returns a distribution on possible results of that experiment (the output of `flip(..)`).

Under m_{bias}, people assume that the coin has some unknown bias, learn it from past observations[4], and use it to predict the next flip:

```
var coinWeights = [0.01, 0.10, 0.20, ..., 0.90, 0.99];
var biasCoin = function(seq) {
  Infer(function(){
    var w = uniformDraw(coinWeights)
    var biasedCoinFlip = function(){ flip(w) }
    var predictedSeq = repeat(seq.length,biasedCoinFlip)
    condition(arrayEquals(seq,predictedSeq))
    biasedCoinFlip()
  })
}
```

Under m_{markov}, people assume that the flips are generated by a Markov process with transition probability `p`, which is learned from past observations:

```
var markovCoin = function(seq1) {
  Infer(function(){
    var p = uniformDraw(coinWeights)
    var markovFlip = function(lastFlip) {
      flip(p) ? !lastFlip : lastFlip
    }
    var sampleSeq = function(n, seqSoFar) {
      if (n == 0) {
        seqSoFar
      } else {
        var nextFlip = markovFlip(last(seqSoFar))
        var nextSeq = append(seqSoFar, nextFlip)
        sampleSequence(n - 1, nextSeq)
      }
    }
    var seq2 = sampleSeq(seq1.length - 1, [flip(0.5)])
    condition(arrayEquals(seq1, seq2))
    markovFlip(last(sampledSeq))
  })
}
```

[4] The line that uses `condition` constrains likely values of the coin weight—this mechanism is used to represent learning in Bayesian models of cognition. For more, see the Learning as Conditional Inference chapter of the online textbook `http://probmods.org`.

Predictions of optimal experiment design

Using a uniform distribution for $p(y;x)$, we ran OED for three different model comparisons: fair–bias, bias–Markov, and fair–bias–Markov and planned to collect data from 20 participants (rather than 1).[5] We run `OED` by writing:

```
var n = 20,
    fairGroup = groupify(fairCoin),
    biasGroup = groupify(biasCoin)
OED({
  M: function() { uniformDraw([fairGroup, biasGroup]) },
  X: function() {
    {n: n, seq: uniformDraw(["HHHH",...,"TTTT"])}
  },
  Y: function(x) { randomInteger(n + 1) }
})
```

We define a uniform prior on models `M`, an experiment space `X` with a fixed number of subjects and all valid coin sequences, and a result space `Y`, which is the uninformative prior over the number of `H` responses. The results of different model comparisons are below:

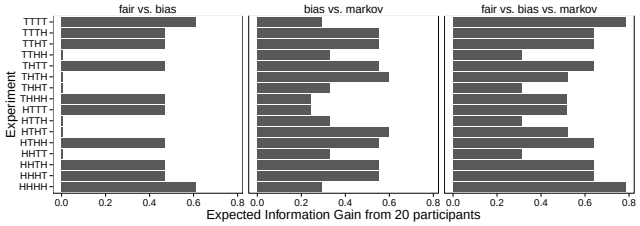

Figure 1: Results for sequence prediction model comparisons

Consider the fair–bias comparison (Fig. 1, left). Several experiments have 0 information gain (e.g., `HTHT`)—the models make exactly the same predictions in this case (albeit for different reasons, as discussed earlier), so the experiment has no distinguishing power. The best experiments are `HHHH` and `TTTT`. This is intuitive—the bias model infers a strongly biased coin and makes a strong prediction, while the fair coin model is unaffected by past observations.

In the bias–Markov comparison (Fig. 1, middle), the best and worst experiments actually reverse. Now, `HHHH` and `TTTT` are the least informative (because, as before, the models make similar predictions here), whereas `HTHT` and `THTH` are the most informative. This makes sense—the bias model learns a weight of 0.5 and so assigns equal probability of heads and

[5] Our models are of a single subject but we lift each single-participant model into a model of group responses using an i.i.d. linking function that we call `groupify`:

```
var groupify = function(model) {
  var groupified = function(x) {
    var sequence = x.sequence, n = x.n;
    var singleModel = model(sequence);
    var p = Math.exp(singleModel.score(true))
    Binomial({n: n, p: p})
  }
  groupified
}
```

Here, `singleModel.score(true)` returns the log-probability of the value `true` under the `singleModel` distribution.

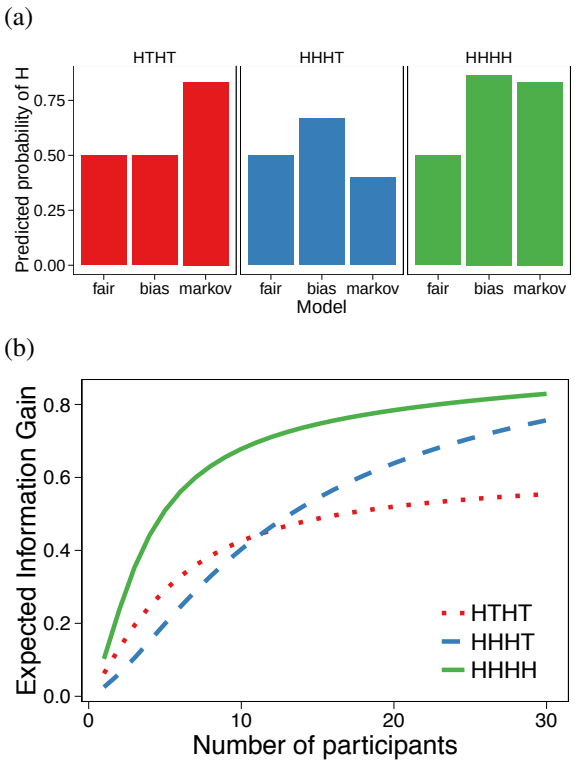

Figure 2: Top three experiments in three-way model comparison: (a) model predictions and (b) EIG versus sample size.

tails to the next flip, whereas the Markov model learns that the transition probability is high and assigns high probability to the opposite of the most recent flip (T for THTH and H for HTHT).

In the full fair–bias–Markov comparison (Fig. 1, right), the worst experiments (e.g., TTHH) are again cases where all models make similar predictions. The best experiments are TTTT and HHHH, a result that is non-obvious because we are comparing three models rather than two. The best experiment HHHH is very good at separating the fair model from the other two models, while still predicting a difference between bias and Markov (Fig. 2a, right). The second best experiment, HHHT, predicts three qualitatively different responses for the three models: bias model above baseline, Markov model below baseline, and fair model at baseline (Fig. 2a, middle), but this comes at the cost of less EIG overall. An automated design tool is especially useful in these settings, where human intuition would likely favor the qualitative over the quantitative differences.

Finally, an experiment's EIG varies as a function of sample size (Fig. 2b). This function is non-linear and, crucially, the rank ordering of experiments can change. For the the full model comparison, the experiments HTHT and HHHT switch places after 12 participants. This is particularly relevant when three models are being compared; small quantitative differences between two models may grow with the sample size.

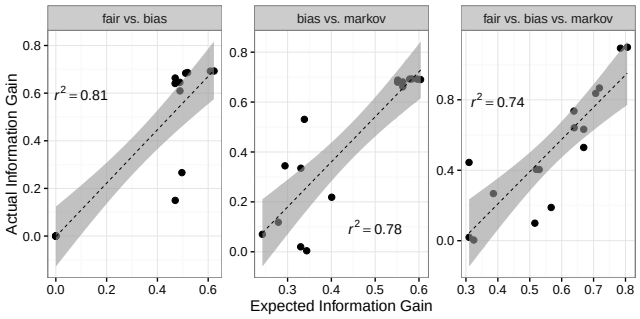

Figure 3: AIG vs. EIG for each experiment

Empirical validation

We validated our system by collecting human judgements for all 16 experiments and comparing *expected* information gain (EIG) with the *actual* information gain (AIG) from the empirical results. We randomly assigned 351 participants to a single experiment (sequence). All of the 16 experiments were completed by $\geq$20 unique participants.[6] Participants pressed a key to sequentially reveal the sequence of 4 flips and then predicted the next coin flip (either heads or tails).

For each of the three model comparison scenarios described earlier, we compared EIG to AIG for every experiment x. Figure 3 shows that EIG is a reliable predictor of AIG—the empirical value of an experiment (minimum r = 0.857). This indicates that the OED tool could be relied on to automatically choose good experiments for this case study.

Case study 2: Category learning

Here, we explore a more complex and theoretically important set of models and experiments. In addition, whereas the previous section considered Bayesian cognitive models, here we consider non-Bayesian models of category learning. In particular, we analyze a classic paper on the psychology of categorization by Medin and Schaffer (1978) that aimed to distinguish two competing models of category learning – the *exemplar model* and the *prototype model*. By hand, Medin and Schaffer (MS) designed an experiment (often referred to as the "5-4 experiment") where the models made diverging predictions and found that the results supported the exemplar model. Subsequently, many other authors followed their lead, replicating and using the 5-4 experiment to test other competing models. Here, we ask: how good was the MS 5-4 experiment? Could they have run an experiment that would have distinguished the two models with less data?

Models

Both the exemplar and prototype models are classifiers that map inputs (objects represented as a vector of Boolean features) to a probability distribution on the categorization response (a label: A or B). The exemplar model assumes people

[6] N's were uneven due to randomization. We use the empirical N's for comparing EIG to AIG.

store information about every instance of the category they have observed; categorizing an object is thus a function of the object's similarity to all of the examples of category A versus the similarity to all of B's examples. By contrast, the prototype model assumes that people store a measure of central tendency for each category—a prototype. Categorization of an object is thus a function of its similarity to the A prototype versus its similarity to the B prototype. For space, we omit these model implementations but refer interested readers to the source code available online.

Experiments

Participants first learn about the category structures in a training phase where they perform supervised learning of a subset of the objects and are then tested on this learning in a test phase. During training, participants see a subset of the objects presented one at a time and must label each object. Initially, they can only guess at the labels, but they receive feedback so that they can eventually learn the category assignments. After reaching a learning criterion, they complete the test phase, where they label all the objects (training set and the held out test set) without feedback.

MS used visual stimuli that varied on 4 binary dimensions (color: *red* vs. *green*, shape: *triangle* vs. *circle*, size: *small* vs. *large*, and count: *1* vs. *2*). For technical reasons, they considered only experiments that (1) have linearly separable decision boundaries, (2) contain 5 A's and 4 B's in the training set, and (3) have the modal A object `1111` and the modal B object `0000`. There are, up to permutation, 933 experiments that satisfy these constraints.

Predictions of optimal experimental design

Using the predictive prior for $p(y;x)$, we computed EIG for all 933 experiments and found that the optimal experiment (for a single participant) sets the As to be `0001, 0011, 1100, 1110, 1111` and the Bs to be `0100, 0110, 1000, 1010`. By contrast, the MS experiment sets the As as `1110, 1010, 1011, 1101, 0111` and the Bs as `1100, 0110, 0001, 0000`. The optimal experiment had an EIG of 0.08 nats while the MS experiment had an EIG of only 0.03 nats, a 2.5-fold difference. Indeed, the MS experiment is near the bottom third of all experiments (Fig. 5a).

Why is the MS experiment relatively ineffective? One reason is that Medin and Schaffer prioritized experiments that predict a qualitative categorization difference. In particular, they argued that the prototype model predicts that object `1110` should be easier to learn than object `1010`, whereas the exemplar model predicts the reverse. However, this qualitative difference between two objects comes at the cost of little information gain from the remaining objects (Fig 4). The optimal experiment better disambiguates the models by maximizing the information from all test objects simultaneously.

Empirical validation

To validate our EIG calculations, we ran the MS and optimal experiment with 60 participants each. Figure 5b shows

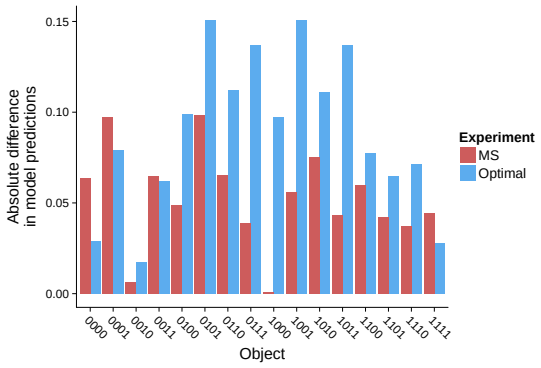

Figure 4: In MS and optimal experiments, the divergence between model predictions (i.e., the absolute value of the difference in probabilities of being classified as an A by the two models). The models diverge more in the optimal experiment.

that the optimal experiment we found for a single participant is indeed better than the MS experiment (n=1, blue greater than red). For n=1, the mean actual information gain (AIG) for the optimal experiment is 0.15, whereas it is 0.026 for the MS experiment. This 5-fold difference is even greater than the 2.5-fold difference predicted by EIG. In addition, by incrementally introducing more data, we observe that both experiments eventually reach maximal AIG but the optimal experiment takes only 10 participants to do so whereas the MS experiment takes around 30 participants. Thus, the optimal experiment could provide the same amount of information for a third of the experimental cost.

Related work

The basic intuition behind OED—to find experiments that maximize some measure of expected informativeness—has been independently discovered in a number of fields, including physics (van Den Berg & Curtis, 2003), chemistry (Huan, 2010), biology (Liepe, Filippi, Komorowski, & Stumpf, 2013; Vanlier, Tiemann, Hilbers, & van Riel, 2012), psychology (Myung & Pitt, 2009), and statistics (Lindley, 1956).

Previous work, however, has either been too narrow for general use or required too much statistical and computational expertise. For example, Liepe et al. (2013) devised a method for finding experiments that optimize information gain for parameters of biomolecular models (ODEs with Gaussian noise). Myung and Pitt (2009) described a more general optimization method but this requires users to select their own utility function for the value of an experiment and implement inference on their own. For example, they compared six memory retention models using Fisher Information Approximation as a utility function and performed inference using a custom annealed SMC algorithm. Such "bring-your-own" requirements impose a significant burden on users and are a real barrier to entry.

By contrast, our OED system is general and practical, which allows users to rapidly explore different spaces of mod-

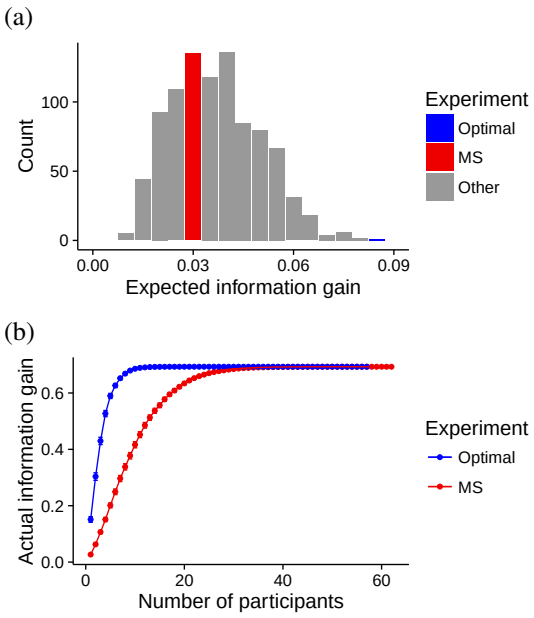

Figure 5: (a) Distribution of EIG for all category learning experiments on a single participant. MS has low EIG. (b) AIG versus number of participants in analysis (error bars are 95% bootstrapped confidence intervals). MS requires more participants to achieve maximum AIG.

els, experiments, and inference algorithms. This approach is compatible with certain challenging features of cognitive science experiments: participants give noisy responses, experimental results are sensitive to sample size, and models often require *linking functions* to convert model output into predictions about experimental data. Additionally, our work is the first to demonstrate that expected information gain is a reliable predictor of actual information gain.

Conclusion

Cognitive scientists aim to design experiments that yield informative results. `webppl-oed` partially automates experiment design, searching for experiments that maximally update beliefs about the model distribution. With our approach, scientists write their models as probabilistic programs, define a space of possible experiments and results, and hand these to OED for experiment selection. We stress that the tool *complements* scientists; it does not replace them. Our tool merely eliminates the need to manually comb large spaces of potential experiments, which is time consuming, tedious, and prone to bias, such as a preference for local qualitative differences at the expense of ultimate quantitative information gain. The real ingenuity—devising empirical paradigms and building models—must still come from the scientist.

Our approach suggests a number of interesting directions for future work. First, OED can be computationally challenging—our software currently does not scale up to huge response or experiment spaces, so there is still room for optimizing search algorithms. Second, we have examined model comparison problems where there are a finite number of models. We believe that our approach also works in (1) parameter learning settings where the goal is to conduct experiments that best update beliefs about continuous parameters of a model, and (2) model comparison problems where a finite number of models each have continuous parameters that are unknown and must be integrated over.

Lastly, we have restricted attention to "one-shot" experiments, but it would be useful to extend our work to sequential settings such as adaptive testing. Adaptive testing can be formulated as a problem of information gain of *sequences* of experiments, which produce dependent and non-iid responses. Some preliminary work suggests that `webppl-oed` can be profitably extended to the adaptive setting.

References

Falk, R. (1981). The perception of randomness. *International Conference for the Psychology of Mathematics Education*, *1*, 222-229.

Goodfellow, L. D. (1938). A psychological interpretation of the results of the Zenith radio experiments in telepathy. *Journal of Experimental Psychology*, *23*(6), 601-623.

Goodman, N. D., & Stuhlmüller, A. (2014). *The Design and Implementation of Probabilistic Programming Languages.* `http://dippl.org`.

Griffiths, T. L., & Tenenbaum, J. B. (2004). From algorithmic to subjective randomness. In *Advances in neural information processing systems.*

Huan, X. (2010). *Accelerated Bayesian experimental design for chemical kinetic models* (Unpublished master's thesis). Massachusetts Institute of Technology.

Liepe, J., Filippi, S., Komorowski, M., & Stumpf, M. P. H. (2013, January). Maximizing the Information Content of Experiments in Systems Biology. *PLoS Computational Biology*, *9*(1), 1-13.

Lindley, D. V. (1956). On a measure of information provided by an experiment. *Annals of Mathematical Statistics*, *27*(4), 986–1005.

Medin, D. L., & Schaffer, M. M. (1978). Context Theory of Classification Learning. *Psychological Review*, *85*(3), 207-238.

Myung, J. I., & Pitt, M. A. (2009). Optimal experimental design for model discrimination. *Psychological review*, *116*(3), 499–518.

van Den Berg, J., & Curtis, A. (2003). Optimal nonlinear Bayesian experimental design: an application to amplitude versus offset experiments. *Geophysical Journal International*, *155*(2), 411–421.

Vanlier, J., Tiemann, C. A., Hilbers, P. A. J., & van Riel, N. A. W. (2012, April). A Bayesian approach to targeted experiment design. *Bioinformatics*, *28*(8), 1136–1142.