Online Appendix

Adaptive Preference Measurement with Unstructured Data

Ryan Dew

April 12, 2024

Contents

A. Generating Embeddings from VGG-19

For our image data, to learn embeddings directly from pixel-level image data, we propose using transfer learning via pre-trained models. The intuition behind this approach is to take a deep learning model that has already been trained for a different purpose (e.g., image classification), feed our images into that model, then extract the model's internal activations to serve as our representations. The specific pre-trained model we use is VGG-19, which was originally used to classify images into 1,000 classes (Simonyan and Zisserman, 2014). The architecture of VGG-19 consists of 19 layers: a series of convolutional layers, meant to extract features from the input images, followed by a series of fully connected layers aimed at synthesizing those features into useful representations of the inputs, and ending with a softmax layer, which converts those representations into probabilities over the 1,000 classes. To appropriate VGG-19 for the purposes of learning image representations, we feed our product images $\mathbf{x}_j$ through each of the layers, stopping at the last fully connected layer before the softmax, which results in a 4,096-dimensional vector. After doing this for all product images, we take the resulting set of J 4,096-dimensional vectors and apply principal components analysis, reducing the dimensionality to a desired level, D. The loadings of each image on the resulting D-dimensions form our representations, $\mathbf{z}_j$.

B. Hyperparameter Optimization

We now describe in more detail several aspects of hyperparameter optimization, including the role the hyperparameters play in effective learning, the objective functions for optimizing the hyperparameters, the specific priors used in the empirical application, and the procedure for aggregate hyperparameter optimization.

B.1. Hyperparameters and Adaptive Learning

Recall that the utility model depends three hyperparameters: the kernel hyperparameters, $\theta_i = \{\eta_i, \rho_i\}$, which capture the amplitude and smoothness of the function respectively, and the noise of the ratings, σ_i^2. These parameters determine not only the eventual properties of $u_i(\mathbf{z})$, but also affect how efficiently we are able to learn about a consumer's preference. We demonstrate the role these parameters play in adaptive selection in Figure A-1, by seeing how they induce variation in the inferred objective function and its uncertainty. In the left panel, we see the ideal behavior: the estimated function smoothly interpolates between the data, and as we move further away from regions with data, uncertainty slowly goes. In contrast, in the middle panel, we set the function's smoothness to be too low (i.e., ρ_i), which results in the function "jumping" between data points, and rapidly expanding uncertainty intervals. Finally, in the right panel, we set the function's smoothness and noise to both be too high: in this case, everything is interpreted as noise, and the function learned is simply flat. In the latter two cases, badly calibrated hyperparameters would lead to undesirable adaptive behavior, where essentially all points are estimated to be equally uncertain. Thus, calibrating these parameters correctly is essential for effective learning.

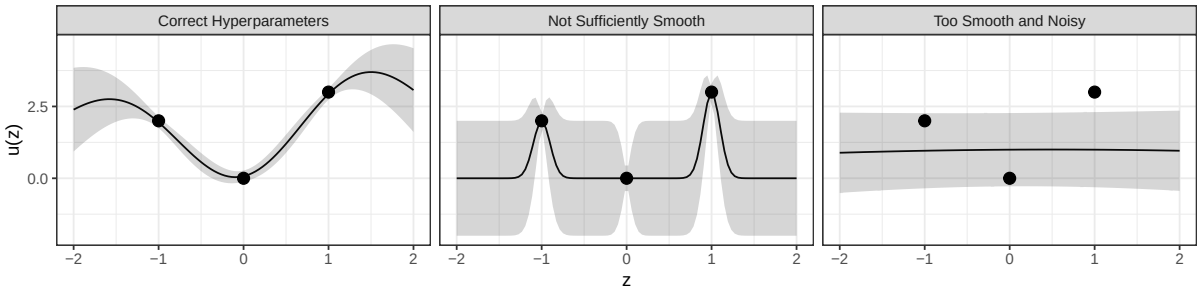


Figure A-1: Role of Hyperparameters
An illustration of the role of the hyperparameters in adaptive learning. At left, an example of properly calibrated hyperparameters. At middle, an example with too low of a lengthscale (ρ_i). At right, an example where the noise (σ_i) is high as well as the lengthscale (ρ_i).

B.2. Hyperparameter Inference

There are two main approaches to hyperparameter inference in GP models: either a fully Bayesian approach, or through maximum marginal likelihood. While a fully Bayesian approach would be ideal, it is also computationally costly, and thus incompatible with our goal of designing an algorithm that can work in real-time, on a simple survey engine. Instead, we use marginal likelihood. The marginal likelihood is the likelihood of the data given the hyperparameters, with the function values marginalized out. The model proposed here is a version of GP regression (Rasmussen and Williams, 2006), and thus, the marginal likelihood can be derived in closed form:

$$\log p(y_i \mid Z_i, \theta_i, \sigma_i) = {y_i}' K_i^{-1} y_i - \frac{1}{2} \log |K_i| - \frac{m}{2} \log 2\pi, \tag{A-1}$$

where $K_i = K(Z_i, Z_i; \theta_i) + \sigma_i^2 I$. To learn good values of θ_i and σ_i, we can maximize Equation A-1 after each data point is acquired, before inferring the function values. This approach is fast, which is needed for an online approach. But, unlike the Bayesian approach, maximizing the marginal likelihood does not, in its standard form, allow for any regularization of the hyperparameters, or any information sharing across people as to reasonable values of the hyperparameters.

To enable on-the-fly optimization, while also accounting for prior information about reasonable hyperparameter values, we augment the marginal likelihood with priors for each of the hyperparameters:

$$\log p(\mathbf{r}_i \mid Z_i, \theta_i, \sigma_i) = \mathbf{r}_i' K_i^{-1} \mathbf{r}_i - \frac{1}{2} \log |K_i| - \frac{N}{2} \log 2\pi + \log p(\theta_i, \sigma_i). \tag{A-2}$$

This modification turns maximizing the marginal likelihood into maximum a posteriori (or MAP) inference, where the prior $p(\theta_i, \sigma_i)$ acts to regularize the values of θ_i and σ_i. For simplicity, we use a fully factorized prior, $p(\theta_i, \sigma_i) = p(\eta_i)p(\rho_i)p(\sigma_i)$. While these priors can be any distribution with positive support, in practice, strong regularization of the hyperparameters tends to help performance. There are two important considerations for setting these priors: we want to avoid very low values (i.e., near zero), which can lead to unstable or unidentified estimates, while also regularizing the values toward reasonable values, where "reasonable" depends on the rating scale used, and the way the $\mathbf{z}$-space is scaled. In our empirical application, we use informative gamma and inverse-gamma priors, which we describe in more detail below.

Alternatively, data from past studies or a pre-test can be used to learn good values for the hyperparameters. A simple approach is to learn a single set of hyperparameters that work well across respondents, in aggregate, for use in future studies. We can do this by standard train/test splitting, followed by optimizing the hyperparameters for predictive validity. We

refer to this method as the *aggregate* hyperparameters approach. While this approach is simple, it has two key benefits: first, it saves computation, by estimating optimal parameters offline, before any study is done. Second, by fixing all hyperparameters to a single value, it greatly stabilizes the inference process.

B.3. Priors

When using marginal MAP for individual-level hyperparameter optimization, an important task is setting the prior, $p(\theta_i, \sigma_i)$. While there are certainly interesting potential interdependencies between the hyperparameters — for instance, very noisy observations can be rationalized either by a very unsmooth utility function and low noise, or a very smooth utility function and high noise — in practice, it is much simpler to specify priors for each parameter independently. Thus, we will only consider priors of the form $p(\theta_i, \sigma_i) = p(\eta_i)p(\rho_i)p(\sigma_i)$.

The role of the prior is to regularize the values of these parameters toward reasonable values, to avoid the pathologies discussed above. To set them reasonably, we need to understand what reasonable values for the hyperparameters actually are. Luckily, under the proposed framework, it is possible to a priori specify reasonable values:

- Noise (σ): Reasonable values for the noise depend on how the ratings are operationalized. In the binary choice studies in the main body of the paper, ratings are either r or $-r$, where we set $r = 0.5$. Thus, the total range of ratings is simply 1. This means that, if the standard deviation of the ratings is 0.5, we would effectively have just random ratings. Using this information, we can set the prior to be a distribution positive but low values, roughly between 0 and 0.5. To that end, in the empirical application, we used a Gamma(5,20) distribution, where 5 is the shape parameter, and 20 is the rate parameter. This distribution is plotted in the left panel of Figure A-2.

- Amplitude (η): Reasonable values for the amplitude also depend on how the ratings are operationalized. The amplitude captures essentially the variability in the function values, excluding the noise. Thus, for the binary choice study, we would want functions that can extend between -0.5 and 0.5. Beyond this consideration, very low values of the amplitude can induce some numerical instability in computing the posterior (by, e.g., making the kernel matrix numerically non-invertible, with zero or very tiny but negative eigenvalues). Thus, rather than the gamma distribution, we use an Inverse-Gamma distribution, which is known to exhibit "boundary avoiding" behavior toward zero. Specifically, we use an Inverse-Gamma(1.5, 3) distribution, where 1.5 is the shape, and 3 is the rate. This distribution is plotted in the right panel of Figure A-2.

- Lengthscale (ρ): Reasonable values for the lengthscale depend on distances between z. One way to think about the lengthscale is that it captures how far apart in z-space two points have to be before their function values are uncorrelated, with higher values of the

lengthscale implying larger distances (and thus more smoothness). Just like the ratings scale is determined by the researcher, the scale of z can also be pre-determined. In our studies, we scaled all z to match the scaling of the recommendation system embeddings provided by the company, which were roughly mean-centered in the $[-1, 1]$ range. In addition to the scaling of z, the lengthscale also affects how the model explores: as shown in the main text, if the length-scale is too large, the estimated utility function becomes roughly constant, which inhibits exploration of the interior of the z-space. Thus, setting the prior to encourage relatively *low* lengthscale values is better. Finally, if ρ is too close to zero, there can again be numerical issues. For these reasons, we again used the same inverse gamma distribution as for η as the prior for ρ.

Another consideration is that, especially for studies with real ratings, different respondents may have different tendencies to rate items low or high on the scale. Consider, for instance, one respondent who, on a 10-point scale, always rates items between 3 and 7, and another who rates items between 1 and 10. Ultimately, we care about relative preference, and hence these different respondent-level scalings are not relevant. For this reason, when we ran the ratings-based studies described in Online Appendix D, we always standardized responses after the training phase to mean zero, variance one. Under this renormalization, we know the maximum variance of the utility function is one, and hence fix the amplitude to one, and re-estimate the lengthscale.

B.4. Aggregate Hyperparameter Optimization

Besides estimating individual-level hyperparameters through marginal likelihood maximization, we can also determine a single set of optimal hyperparameters using data from a pre-test. In this case, we gather data from a small set of respondents, showing items either at random, or by using the marginal likelihood method. Then, offline, we optimize the values of the hyperparameters across respondents to maximize model performance in the test set. This procedure

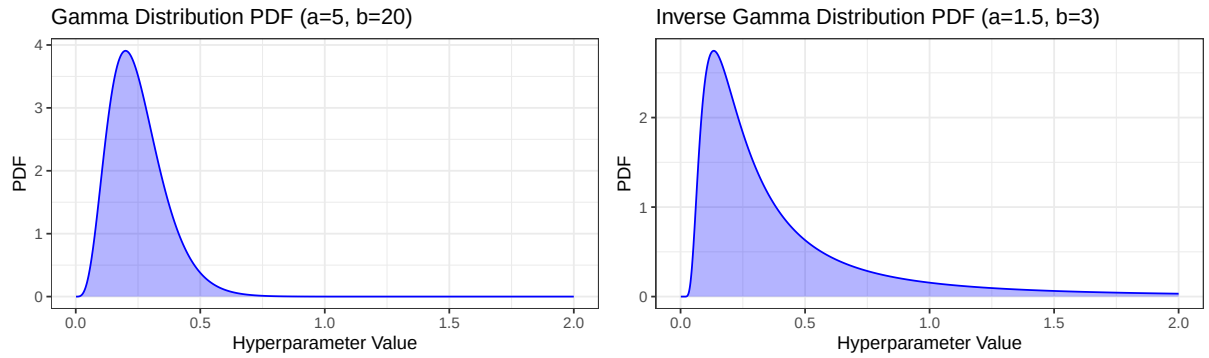


Figure A-2: Probability density functions (PDFs) of the two priors used for the binary choice hyperparameters.

is exactly what is captured by our MU-Agg model.

To actually optimize the hyperparameters, there are several practical concerns, most crucial of which is which objective to actually optimize. For binary choice, accuracy is the most relevant metric. For ratings-based studies, RMSE or correlation between the predicted and actual ratings are more relevant. In practice, however, even for binary choice data, maximizing accuracy may be challenging: for instance, in our application, it was difficult for the optimizer to converge when optimizing for average accuracy. Optimizing for RMSE or correlation proved more robust, and the average test accuracy under the correlation objective slightly outperformed RMSE. Thus, in all studies, we optimized hyperparameters using the correlation between predicted and actual ratings as the objective.

C. Uncertainty Reduction Acquisition Functions

In this appendix, we do two things: first, we introduce the global uncertainty reduction acquisition function (or GUR), which is a direct way of computing the total uncertainty reduction given a new observation. Then, through a simulation, we show how the KUR acquisition function used in the main text relates to GUR.

C.1. Global Uncertainty Reduction (GUR)

The KUR acquisition function is motivated by the idea that we would like to select points that minimize not only uncertainty at a single point, but uncertainty over the entire set of possible items. While KUR does this by means of the estimated kernel function, we can also do this directly, through what we term the Global Uncertainty Reduction, or GUR, acquisition function, given by:

$$a_{\text{GUR}}(z_j) \equiv \sum_{z_{j'} \notin Z_{:(n-1)}} \bar{s}\left(z_{j'} \mid Z_{:(n-1)}\right) - \bar{s}\left(z_{j'} \mid \{Z_{:(n-1)}, z_j\}\right). \tag{A-3}$$

where $\bar{s}(\cdot)$ is the uncertainty around the utility function, which can be derived in closed form as:

$$\bar{s}_i(z \mid Z_i) = k_i(z,z) - K_i(z, Z_i)[K_i(Z_i, Z_i) + \sigma_i^2 \mathbb{I}]^{-1} K_i(Z_i, z). \tag{A-4}$$

Intuitively, the first term of $a_{\text{GUR}}(z)$ is the sum of the posterior variance of the utility function over all possible query points, given we have already observed ratings at points $Z_{:(n-1)}$. The second term is the same, but assuming we additionally observe a rating at z_j. Thus, the first term minus the second term is the reduction in total uncertainty, having observed a rating at z_j.[1] Note that, importantly, $\bar{s}(\cdot)$ does *not* depend (directly) on the actual value of the rating (y_i). Rather, it depends only on what inputs have been observed previously (Z_i), and the kernel, whose hyperparameters may depend on past ratings. Thus, conditional on a set of kernel hyperparameters, this acquisition function can be hypothetically evaluated, without actually observing the rating at the new point z_j.

The GUR acquisition function is intuitive, but also impractical: the second term involves computing $\bar{s}_i$ under many hypothetical scenarios, which in turn involves matrix inversion, which is of $O(n^3)$ complexity. Importantly, this matrix inversion cannot be stored across iterations of the sum: the actual matrix being inverted changes each time, as we consider a new hypothetical value of z_j. Even for a few hundred items, computing $a_{\text{GUR}}(z_j)$ takes several

[1] GUR can also be extended to explicitly capture symmetry in pairwise studies. The extension is simple: now, when we consider uncertainty reduction, we consider uncertainty reduction given the addition of two new (symmetric) points:

$$a_{\text{GUR-Sym}}(z_p) \equiv \sum_{z_{p'} \notin Z_{:(n-1)}} \bar{s}\left(z_{p'} \mid Z_{:(n-1)}\right) - \bar{s}\left(z_{p'} \mid \{Z_{:(n-1)}, z_p, -z_p\}\right). \tag{A-5}$$

seconds, which is infeasible for real-time use on a survey platform, or in any kind of live on-boarding context. This infeasibility motivated our development of KUR, and our use of KUR in the main text.

C.2. Simulation: GUR vs. KUR

To demonstrate how GUR relates to KUR and its various approximations, we conducted a simulation. In the simulation, the underlying utility is assumed to come from a parametric utility function, with some noise. The specific function used is called the Ackley function, which is a widely used function for testing the performance of optimization algorithms (Molga and Smutnicki, 2005). In one dimension, it is given by:

$$f(\mathbf{x}) = -a\ \exp\left(-b|x_i|\right) - \exp\left(cos(c\ x_i)\right) + a + \exp(1) \tag{A-6}$$

We assumed that z varied from -1 to 1, and used the default values for a, b, and c from the `NiaPy` python package (Vrbančič et al., 2018). This function is useful as it exhibits many local modes, and can be generalized to higher dimensions. We randomly drew 2,000 values of z to serve as the possible query points. The true utility values (i.e., $u(z) + \epsilon$) at those points are given by the blue points in the top panel of Figure A-3.

Given this utility, and the possible query points, the simulation proceeded by mirroring how the adaptive preference measurement procedure would work, assuming a symmetric utility function. To initialize the algorithm, we drew two points at random, and evaluated the utility, for both those two points, and their symmetric points. Those are the black dots in the top panel of Figure A-3. Then, based on the estimated utility function and its uncertainty, shown by the black line and grey error bars, we evaluated the symmetric versions of three acquisition functions: (1) GUR, (2) KUR, and (3) KUR with 1,000 randomly drawn test points, and 1,000 possible candidate points chosen by the MU heuristic (labeled KUR - MU Heur). The acquisition functions themselves (scaled to [0,1] for comparison) are plotted in the bottom panel of Figure A-3, and the z values they each would query next are given by the vertical lines in the top panel.

From this example, we see that the three acquisition functions are very similar: while KUR varies less over z than GUR, their high and low points exactly coincide. KUR with the MU heuristic is essentially pieces of KUR, with the parts of z that were not selected as candidate query points given a zero value. While KUR - MU Heur very closely matches KUR, it is not exactly equal to it: the slight differences are due to the random selection of test points. Finally, note that the MU heuristic did not miss any relevant areas identified by GUR or KUR without the heuristic: all three acquisition functions suggested essentially the same optimal next query.

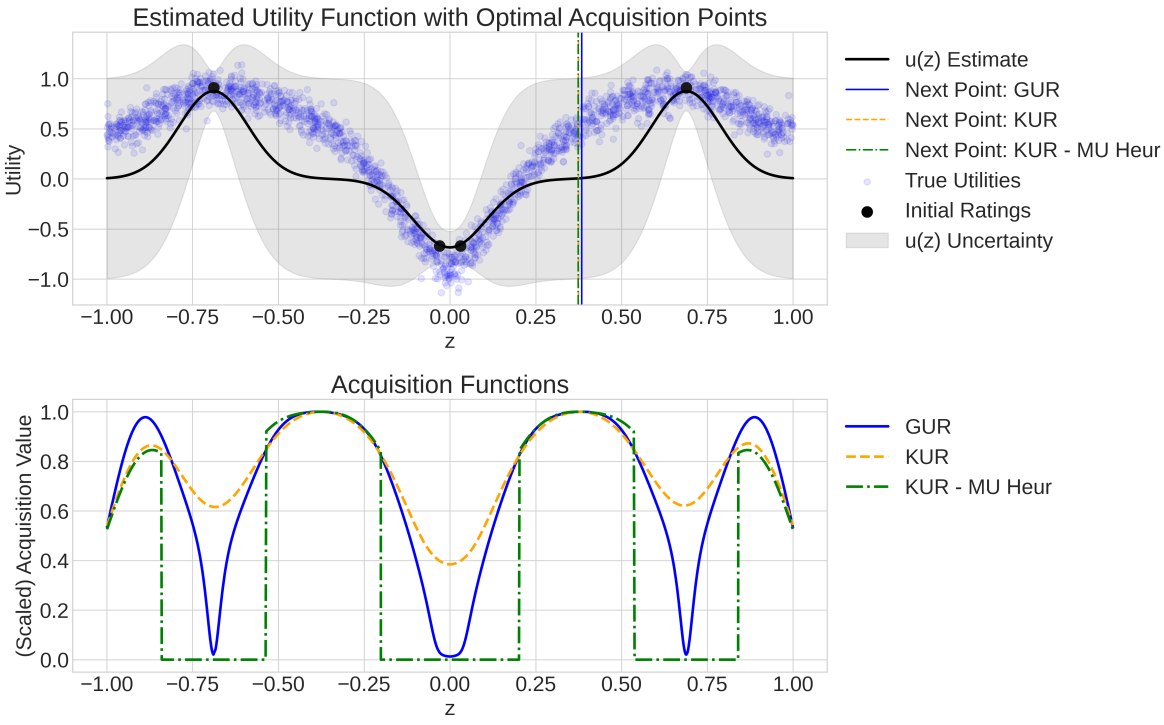


Figure A-3: Illustration of the three GUR-style acquisition functions on synthetic data.

D. Ratings-based Studies

In this appendix, we summarize the results for a series of studies using ratings-based questions. In the ratings tasks, we asked respondents to rate a single item, rather than evaluate pairs of items. Respondents were prompted to rate the item on a scale from 0-10, with higher ratings corresponding to higher preference. An example of a ratings-based task is shown in Figure A-4.

Across a series of studies, we examined the performance of the following algorithms, most of which are the same as in the binary choice tasks from the main body of the manuscript:

1. **Nearest:** Pairs of items are shown to respondents at random (i.e., non-adaptively) during the training phase. Predictions for new pairs are made by a nearest neighbor rule: given prior ratings y_n for pairs z_n, $n = 1, \ldots, N$, the prediction for a new pair z' is $\hat{y} = y_n$ for the n such that $||z' - z_n||$ is smallest. This model mirrors typical collaborative filtering approaches.

2. **MU:** The GP-based model with individual-level hyperparameters learned by MAP (i.e., Equation A-2). Items are selected using the MU acquisition function.

3. **MU-Agg:** The same as MU, but with aggregate hyperparameters learned from a pre-test. Specifically, the pre-test was the random item data gathered from the Nearest study.

4. **KUR:** The same as MU, but with the KUR acquisition function.

We do not consider Abernethy et al.'s adaptive linear model, as it is defined only for metric paired comparisons. For all models, the item embeddings were from the company's recommendation system ("RecSys").

The studies were all conducted exactly as the binary choice studies in the main text, including a training phase of 40 ratings, and a test phase consisting of 10 randomly chosen unseen items to rate, and 5 recommendation items. However, since the outcome is now a rating, rather than a choice, the relevant evaluation metrics are slightly different. In particular, we consider two metrics: the correlation between the predicted and actual ratings, and the root mean squared error (RMSE) between the predicted and actual ratings. In practice, correlation is the more relevant metric: we only care about rank ordering the items, not necessarily about predicting precisely what rating the person would give. Thus, even if the actual numeric prediction is slightly off (as is often the case if, e.g., we impose strong regularization on the hyperparameters), as long as more preferred items have higher predicted utility, all of the insights from the framework will be correct.

Similar to the binary choice studies, as part of the test phase, there was also a "recommendation" task. In this case, the model predicted which five of the previously unseen items the respondent would like the most, which were then shown to the respondent. Intuitively, this

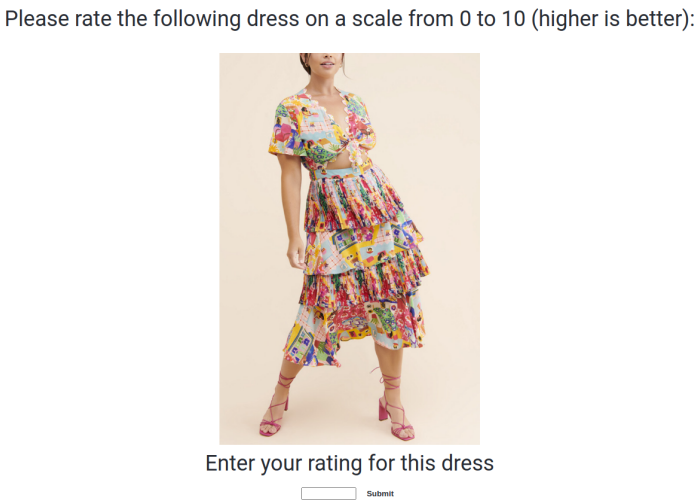


Figure A-4: An example of a rating task from the application.

task mirrors the recommendation context, where we only need to identify preferred items, not make predictions for a random set of items. To evaluate the success of this, we considered an accuracy measure, defined as whether the respondent rated these recommended items above (or below) their training phase average rating.

The average results across models, including both the random item tasks, and the recommendation tasks, are shown in Table A-1. We see that the results are largely in line with the binary choice results: the proposed methods outperform the simple nearest neighbor method. Among them, MU-Agg performs best, which makes sense, given it uses pre-test information. Interestingly, KUR did not perform as well for the ratings-based task as it did for the binary choice, relative to the other acquisition functions.

We see this same pattern of results even more strongly when we consider individual-level statistics. In Figure A-5, we plot the distribution of individual-level results for the correlation in the random item test, and the accuracy for the recommendation task. Echoing what we saw in the binary choice task in the main text, we see the adaptive models work well, achieving above zero correlation for nearly all respondents. Moreover, for recommending items, for the vast majority of respondents, the model correctly recommended good items at least 4 of 5 times.[2]

[2]Because of an implementation error, there was no recommendation-style task for the Nearest model. Thus, we do not present the recommendation results like in binary choice or ratings.

Model	Random Items Corr.	RMSE	Recom. Accuracy
Nearest	0.173	0.544	0.660
	(0.013)	(0.008)	(0.010)
MU	0.349	0.407	0.804
	(0.014)	(0.006)	(0.010)
MU-Agg	0.353	0.389	0.818
	(0.015)	(0.006)	(0.010)
KUR	0.243	0.373	0.718
	(0.034)	(0.015)	(0.023)

Table A-1: Prediction Results for Ratings-based Studies
Individual-level results for each model. The first and second columns are the prediction statistics for rating a random set of items. The third column is the accuracy for the recommendation-like tasks. Standard errors are in parentheses.

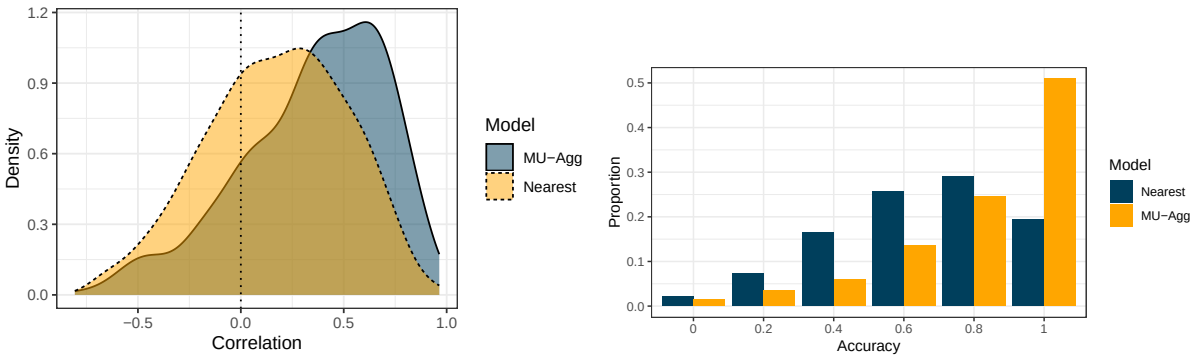


Figure A-5: Individual-level Results
A comparison between the best performing adaptive model, MU-Agg, and the baseline benchmark Nearest, showing how the distributions of individual-level results varied across the two. At left, we show the correlation metric for random items. At right, we show the accuracy for the five item recommendation task.

E. Metric Paired Comparisons Studies

Beyond binary choice, it is also possible to extend the proposed framework to handle metric paired comparisons (MPC). In MPC, respondents are shown pairs of items, and for each pair, are asked to rate the extent they prefer one option over the other, typically with a standard Likert-style scale (e.g., "I strongly prefer the left over the right"). An example is shown in Figure A-6. Similar to the binary choice task, this rating is relative: we code the rating scale such that negative values imply item 1 (left) is preferred to item 2 (right), and positive values imply item 2 (right) is preferred to item 1 (left). The magnitude of that rating corresponds to the strength of that preference. If the scale allows for indifference, we assume this corresponds to 0.

In this appendix, I describe the results from three studies run using metric paired comparisons tasks. The studies followed the same flow as the binary choice tasks in the paper, but where the questions were in the form of true metric paired comparisons. As in Figure A-6, all responses were given on a five-point scale, where -2 indicated the person strongly preferred the left item, 0 indicated the person was neutral between the items, and 2 indicated the person strongly preferred the right item. The models considered here were Nearest, the proposed adaptive GP model with the MU acquisition function, and the same but with KUR acquisition.[3]

Performance was evaluated using both the random pair accuracy from the binary choice study, and using the rating-based prediction metrics from the ratings-based studies. For the accuracy metric, one important difference is that indifference was one of the scale items, but the model will never predict exactly zero. Thus, we made the simplifying assumption that indifferent counts as liking both items. This assumption means the accuracy metrics may be somewhat inflated.

The key results are summarized in Table A-2. We see that, broadly, the results are consistent with the binary choice studies: the adaptive methods outperform the random choice, collaborative filtering-style nearest neighbor model. Focusing on the accuracy metrics, we see that the numbers are, on average, higher than for the binary choice studies. The most likely explanation for those gains is the way we treated indifference, which will somewhat artificially induce more correct predictions. At the individual-level, the results are also consistent with the binary choice and ratings-based results, as shown in Figure (again with a slight upward bias), where we plot the distributions of individual fit statistics, for both accuracy and correlation.

[3]While the Abernethy et al. adaptive linear model could hypothetically be used for MPC-style tasks, it was not used in that context in the original paper. Part of the difficulty is that the model fails to initialize when respondents are indifferent to the first few items, which is an issue we encountered in our empirical studies. Hence, we do not benchmark against it here.

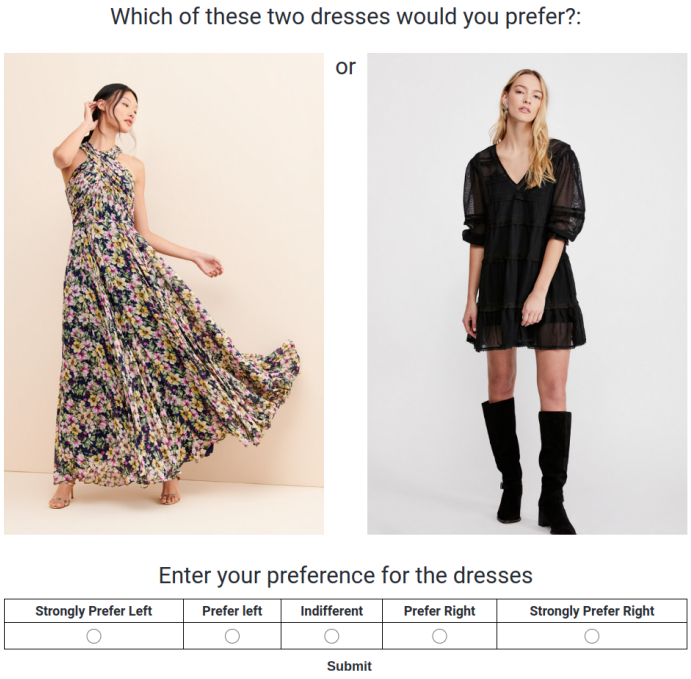


Figure A-6: An Example MPC Task

Model	Binary Prediction Accuracy	Binary Prediction Above 0.5	Rating Prediction Corr	Rating Prediction RMSE
Nearest	0.617	0.672	0.160	0.818
	(0.008)	(0.021)	(0.015)	(0.011)
MU	0.736	0.843	0.375	0.592
	(0.016)	(0.033)	(0.031)	(0.019)
KUR	0.719	0.840	0.332	0.607
	(0.015)	(0.033)	(0.029)	(0.018)

Table A-2: Random Pair Prediction Statistics for MPC Studies
Individual-level results for each model, for the task of predicting preference in random pairs. Accuracy is predicting which item will be preferred, while Corr (for correlation) and RMSE (for root mean squared error) are for predicting the rating itself. Standard errors are in parentheses.

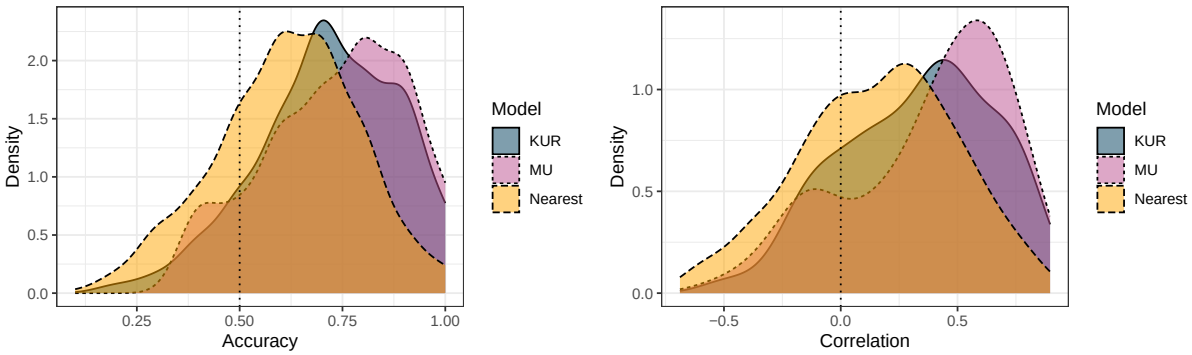


Figure A-7: MPC Individual-level Test Statistic Distributions
At left, the distribution of individual-level accuracies for predicting which of the two items would be preferred. At right, the distribution of individual-level correlations between predicted and actual ratings.

F. Other Embeddings

In the main text, we compared the performance of embeddings learned from the recommendation system to embeddings learned from transfer learning. In this appendix, we consider several modifications to those embeddings. Specifically, we consider four alternative ways of representing items:

- **VGG-PCA-10:** This approach is the same transfer-learned approach described in the main text, by applying VGG-19, extracting the last-before-softmax layer, and using PCA to reduce the dimensionality to 10.
- **VGG-PCA-20:** The same as VGG-PCA-10, but reducing the dimensionality to 20 dimensions instead of 10. Since the original VGG dimensionality is 4,096, we wanted to see whether reducing to just 10 dimensions is too restrictive.
- **PCA-5:** This is the original RecSys embeddings, but reduced to 5-dimensions by PCA. In the cluster analyses presented in the main text, we saw that items could be meaningfully reduced even to 2-dimensions. Hence, we explored the effect of taking the original dimensionality, and reducing it further, via PCA.
- **UMAP-5:** The same as PCA-5, but using UMAP. This is a higher dimension representation than the 2-dimensional representations in text, but lower than the 10-dimensional original RecSys embeddings.

We measured the effectiveness of each embedding in an offline analysis, just like the offline analysis for VGG-PCA-10 in the paper, where we retrained the utility model but using different z values. In this section, we trained the individual-level hyperparameter model for each representation (i.e., no pre-test), using the adaptively chosen items from the original MU study. Because these analyses were done offline, they can be viewed as something of a lower bound on the true performance; if the embeddings were used in a live test, their representation-specific adaptivity could lead them to select items that are even more informative under the metric induced by that representation. We present the results, in the form of learning rate curves, in Figure A-8.

From these learning rates, we can see that the original representations always perform best, which is not surprising: they are supervised recommendation system embeddings. Interestingly, while UMAP is useful as a way of visualizing the results, at least in offline analyses, using it to lower the dimensionality negatively impacts performance. UMAP is designed to represent items in a visually meaningful way, and thus, tends to induce clustering and generally non-uniform distributions over the embeddings space. This non-uniformity could prohibit effective learning under the GP utility model. PCA-5 performed even worse, suggesting if lower dimensionality is needed, UMAP is a better choice, but that 10 dimensions may be

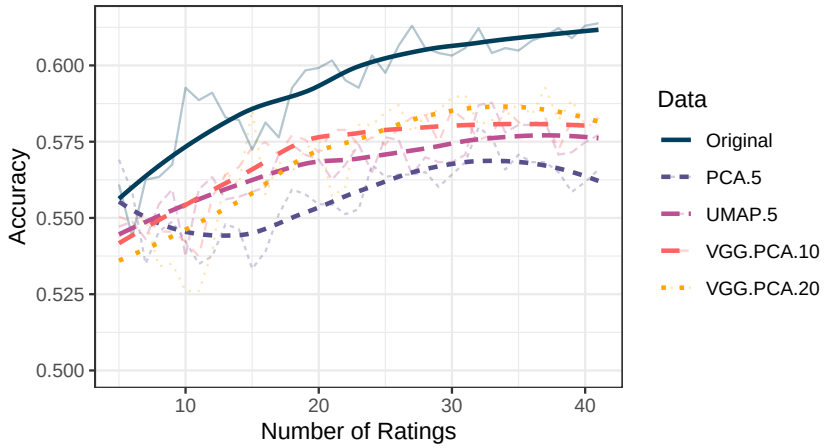
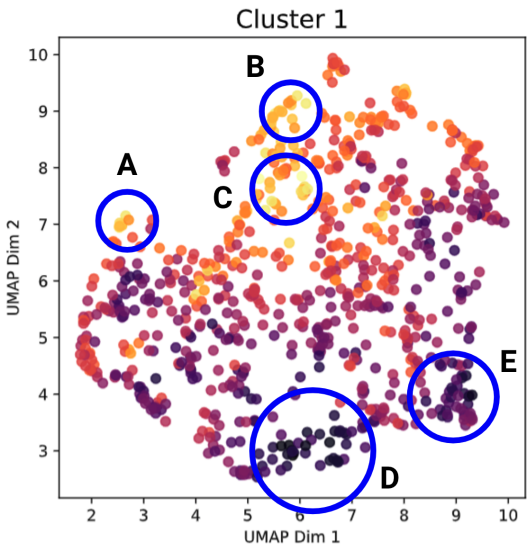


Figure A-8: Offline Analysis of Different Representations
The average accuracy, given a certain number of observations, for individual-level hyperparameter utility models trained using different representations.

the lower bound for meaningfully representing the RecSys embeddings. In terms of transfer learning, we see that the VGG-PCA-10 embeddings and VGG-PCA-20 embeddings perform comparably.

G. Additional Cluster Descriptions

In Figures A-9 through A-14, I show how we can interpret each cluster's UMAP heatmap by examining regions of particularly high or low utility, and looking at exemplar dresses. In each case, example dresses were identified manually, by first identifying polarizing regions of the space (i.e., very high or very low $u(z)$), then looking at the images of items in that region. Two representative items were selected and shown in the table below. In each cluster mean heatmap, the color scheme is consistent with the cluster centers presented in the main body of the paper, with darker colors representing lower preference, and lighter colors representing higher preference. We also provide an interactive version of the UMAP results, where individual items can be seen by hovering over a point, here: `https://adpref.shinyapps.io/umap_explorer/`.



High Utility	Low Utility
Region A	Region D
Region B	Region E
Region C	

Figure A-9: Cluster 1 Mean and Examples
At left is the cluster 1 mean heatmap with labeled regions of interest. At right are example dresses from each region.

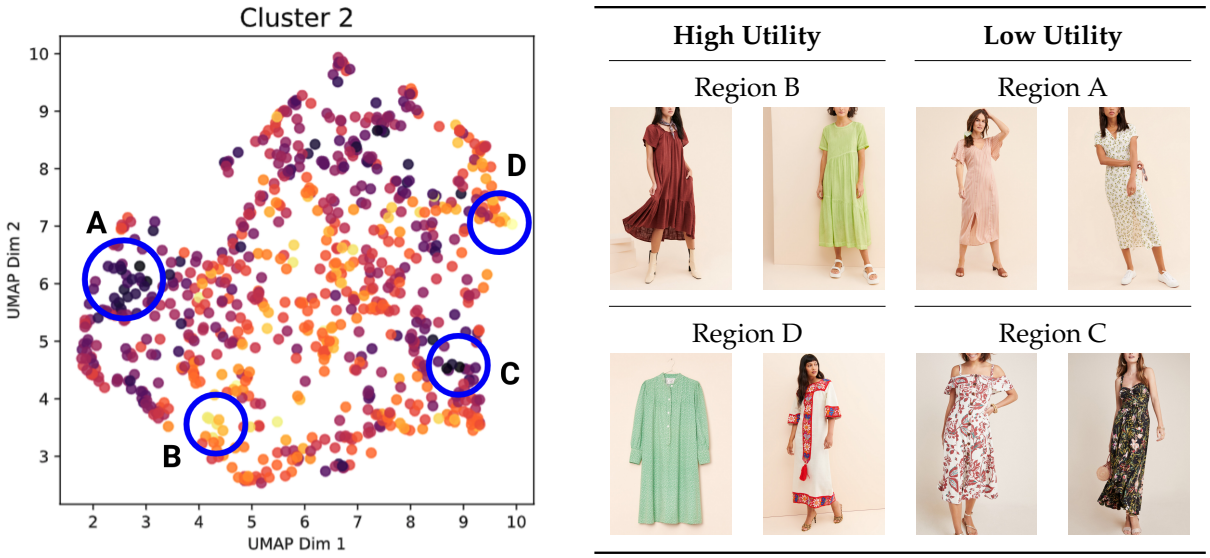


Figure A-10: Cluster 2 Mean and Examples
At left is the cluster 2 mean heatmap with labeled regions of interest. At right are example dresses from each region.

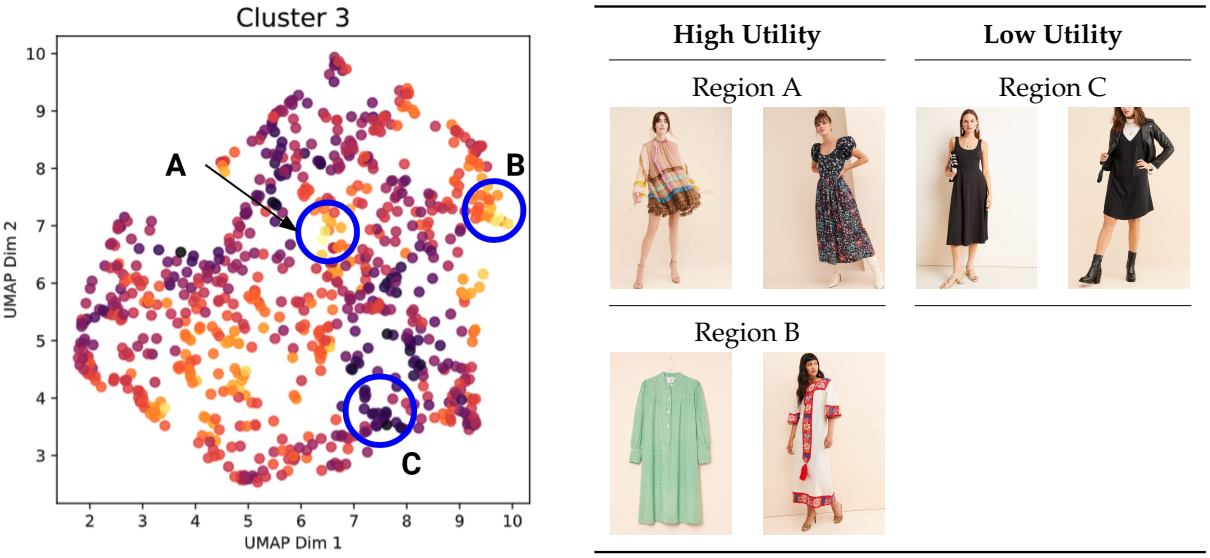


Figure A-11: Cluster 3 Mean and Examples
At left is the cluster 3 mean heatmap with labeled regions of interest. At right are example dresses from each region.

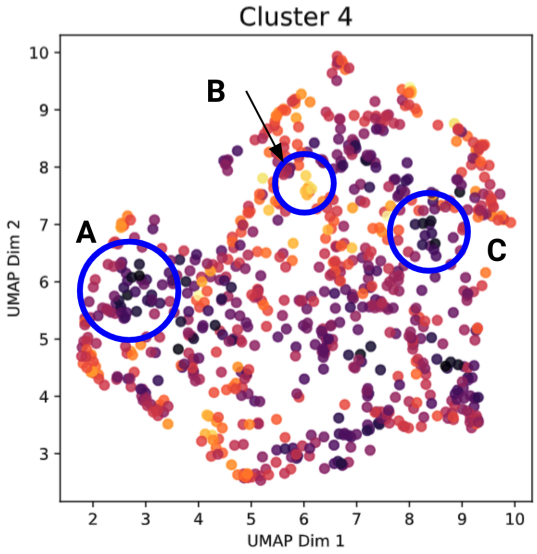


Figure A-12: Cluster 4 Mean and Examples
At left is the cluster 4 mean heatmap with labeled regions of interest. At right are example dresses from each region.

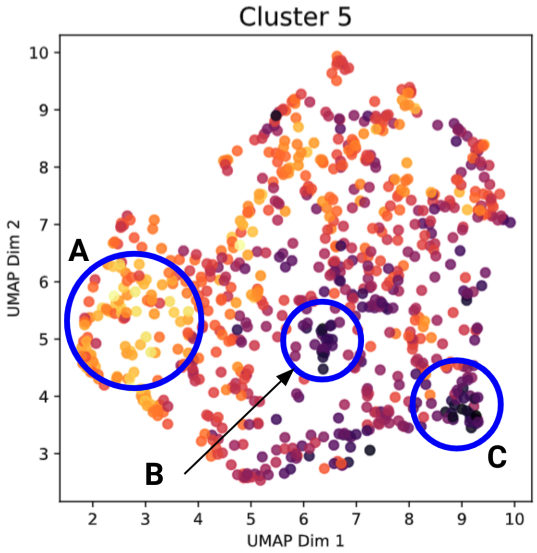


Figure A-13: Cluster 5 Mean and Examples
At left is the Cluster 5 mean heatmap with labeled regions of interest. At right are example dresses from each region.

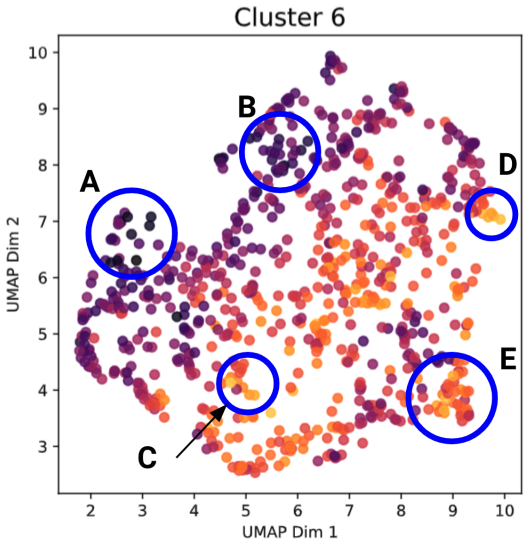


Figure A-14: Cluster 6 Mean and Examples
At left is the Cluster 6 mean heatmap with labeled regions of interest. At right are example dresses from each region.

References

Abernethy, J., Evgeniou, T., Toubia, O., and Vert, J.-P. (2007). Eliciting consumer preferences using robust adaptive choice questionnaires. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):145–155.

Molga, M. and Smutnicki, C. (2005). Test functions for optimization needs. Class Notes.

Rasmussen, E. and Williams, K. I. (2006). *Gaussian processes for machine learning*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Vrbančič, G., Brezočnik, L., Mlakar, U., Fister, D., and Fister Jr., I. (2018). NiaPy: Python microframework for building nature-inspired algorithms. *Journal of Open Source Software*, 3.