Dynamic Question Ordering: 
Obtaining Useful Information While Reducing User Burden

Proposal

Kirstin Early

October 13, 2016

Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Jennifer Mankoff, Co-chair
Stephen Fienberg, Co-chair
Jared Murray
John Abowd

Submitted in partial fulfillment of the requirements for the degree
of Doctor of Philosophy in Machine Learning

©Kirstin Early, 2016
Abstract

Collecting data from individuals can be useful to the individuals (by providing them with personalized predictions) and the data collectors (by providing them with information about populations). However, collecting these data is costly: answering survey items, collecting sensed data, and computing values of interest deplete finite resources of time, battery life, money, etc. Dynamically ordering the items to be collected, based on already known information (such as previously collected items or paradata), can lower the costs of data collection by tailoring the information-acquisition process to the individual. This thesis presents a framework for an iterative dynamic item ordering process that trades off item utility with item cost at data collection time. The exact metrics for utility and cost are application-dependent, and this framework can apply to many domains. The two main scenarios we consider are (1) data collection for personalized predictions and (2) data collection in surveys. We illustrate applications of this framework to multiple problems ranging from personalized prediction to questionnaire scoring to government survey collection. We compare data quality and acquisition costs of our method to fixed order approaches and show that our adaptive process obtains results of similar quality at lower cost.

For the personalized prediction setting, the goal of data collection is to make a prediction based on information provided by a respondent. Since it is possible to give a reasonable prediction with only a subset of items, we are not concerned with collecting all items. Instead, we want to order the items so that the user provides information that most increases the utility to the prediction quality, while not being too costly to provide. One metric for quality is prediction certainty, which reflects how likely the true value is to coincide with the estimated value. Depending whether the prediction problem is continuous or discrete, we use prediction interval width or predicted class probability to measure the certainty of a prediction. We illustrate the results of our dynamic item ordering framework on tasks of predicting energy costs, student stress levels, and device identification in photographs and show that our adaptive process achieves equivalent error rates as a fixed order baseline with cost savings up to 45%.

For the survey setting, the goal of data collection is often to gather information from a population, and it is desired to have complete responses from all samples. In this case, we want to maximize survey completion (and the quality of necessary imputations), and so we focus on ordering items to engage the respondent and collect hopefully all the information we seek, or at least the information that most characterizes the respondent so imputed values will be accurate. One item utility metric for this problem include information gain to get a “representative” set of answers from the respondent. Furthermore, paradata collected during the survey process can inform models of user engagement that can influence either the utility metric (e.g., likelihood the respondent will continue answering questions) or the cost metric (e.g., likelihood the respondent will break off from the survey). We illustrate the benefit of dynamic item ordering for surveys on two nationwide surveys conducted by the U.S. Census Bureau: the American Community Survey and the Survey of Income and Program Participation.
Chapter 1

Introduction

Online data collection from individuals can provide them with personalized predictions (e.g., recommender systems), in addition to gathering information from populations of interest (e.g., surveys), at scale and at low cost to the data collectors. However, users do not have the resources to provide all the information we seek—they cannot answer too many questions that may be difficult to get answers for, and their mobile devices do not have sufficient battery life to provide constant streams of high-fidelity sensed data. Strategically choosing which feature to obtain next from a particular user, depending on previous responses, can lower the burden on respondents while still collecting useful information. This idea of dynamic question ordering (DQO) can also take into account the varying costs of individual questions—a question can be costly to answer due to the effort required for a respondent to come up with an answer, the drain on a battery for sensing a certain measurement, or the likelihood that a respondent will break off from a survey when presented with the question (among other types of cost). DQO trades off the utility we get from having an answer due to the effort required for a respondent to come up with an answer, the drain on a battery for sensing a certain measurement, or the likelihood that a respondent will break off from a survey when presented with the question (among other types of cost). DQO trades off the utility we get from having an answer to a question with its cost and sequentially requests feature values in order to make useful, confident predictions and gather survey data with the resources users are willing and able to provide.

We propose a general framework for dynamically ordering questions, based on previous responses, to engage respondents, improving prediction quality and survey completion. Our work considers two scenarios for data collection from survey-takers. In the first, we want to give the respondent a personalized prediction, based on information they provide. Since it is possible to give a reasonable prediction with only a subset of questions, we are not concerned with motivating the user to answer all questions. Instead, we want to order questions so that the user provides information that most reduces the uncertainty of our prediction, while not being too burdensome to answer. In the second scenario, our goal is to maximize survey completion (and the quality of necessary imputations) and so we focus on ordering questions to engage the respondent and collect hopefully all the information we seek, or at least the information that most characterizes the respondent so imputed values will be accurate.

1.1 A note on vocabulary

This idea of dynamic question ordering can be applied to a broad number of applications and domains, and the vocabulary is often specific to each domain, with different terms capturing similar concepts across applications. In this thesis, rather than enforcing a standard vocabulary across all domains, we will adopt the domain-specific terminology for each application, keeping in mind that the specific word refers to a more general idea. Table 1.1 summarizes these equivalent terms, according to which main domain (prediction or survey-taking) they appear in.
Table 1.1: Equivalent words used for different applications in this thesis.

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Prediction-focused</th>
<th>Survey-focused</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information to be collected</td>
<td>Feature</td>
<td>Question</td>
</tr>
<tr>
<td>Time of data collection</td>
<td>Test time</td>
<td>Survey time</td>
</tr>
<tr>
<td>Information provider</td>
<td>User/system</td>
<td>Respondent</td>
</tr>
<tr>
<td>Information cost</td>
<td>Cost</td>
<td>Burden</td>
</tr>
</tbody>
</table>

1.2 Thesis statement and research questions

Dynamically ordering questions, based on already known information (such as previously answered questions or paradata), can lower the costs of data collection and prediction by tailoring the information-acquisition process to the individual. This thesis presents a framework for an iterative dynamic question ordering process that trades off question utility with question cost at data collection time; the exact metrics for utility and cost are application-dependent. We compare data quality and acquisition costs of our method to fixed order approaches and show that our adaptive process obtains results of similar quality at lower cost.

1.2.1 Research questions

To explore and validate the promise of this thesis, we consider 8 specific research questions (RQs).

- **RQ 1**: How can we measure the “usefulness” (utility) of acquiring a piece of information, in the context of what information has already been acquired? To what extent is this metric application-dependent?

- **RQ 2**: How can we measure the cost of each piece of information? To what extent is this metric application-dependent?

- **RQ 3**: How do prediction quality and cost evolve as more information is obtained?

- **RQ 4**: To what extent does dynamic question ordering change the order used for each sample?

- **RQ 5**: How do context-dependent costs influence the question orders resulting from various contexts?

- **RQ 6**: How can population-level estimates be obtained from dynamically ordered questionnaires?

- **RQ 7**: In the survey-taking setting, how can user engagement be modeled and influence the question ordering process?

- **RQ 8**: How can this statistics- and machine-learning-focused approach incorporate qualitative work in survey methodology; e.g., results on questions’ order effects and cognitive burden?

1.3 Outline

The remainder of this proposal is organized as follows: Chapter 2 covers previous work for this problem, in the areas of machine learning and survey methodology. Chapter 3 identifies what gaps remain in the past work and presents a general framework for dynamic question ordering. Chapters 4 and 5 tailor this framework to situations of prediction and survey-taking, respectively, along with example applications in domains such as energy, health, ubiquitous computing, and government surveys. Finally, Chapter 6 summarizes which parts of Chapters 4 and 5 remain as future work and contains a timeline to schedule completion of this remaining work.
Chapter 2

Related work

There is a rich literature focusing on adaptively ordering acquisition of information to improve outcomes while minimizing costs, across multiple fields. Examples include test-time feature selection in machine learning, adaptive testing in educational research, and adaptive survey design in survey methodology.

2.1 Test-time feature acquisition

In standard supervised machine learning approaches, a predictor is learned from training data and used to make a prediction on a test point. It is assumed that the training data and test points share a common set of features, but labels are provided for only the training data.

Because labels can be costly to acquire at training time, a large body of related work has focused on active learning, which strategically selects which unlabeled data to acquire labels for, so as to maximize a model’s performance while minimizing the cost of data collection (Cohn, Ghahramani, & Jordan 1996).

Alternatively, it may be features that are costly to acquire at test time (or, equivalently, prediction time, for deployed systems). For application areas where feature computation time is a bottleneck at test time (e.g., natural language processing (NLP), computer vision), the primary goal of test time feature acquisition is to speed up prediction (e.g., (He, Daumé III, & Eisner 2013; Strubell, Vilnis, Silverstein, & McCallum 2015; Weiss & Taskar 2013)). In other domains, there is an allowable test time budget of costs other than time, such as user burden of providing information (e.g., (Early, Mankoff, & Fienberg 2016; Early, Fienberg, & Mankoff 2016)) or the resulting loss in privacy from disclosing information (e.g., (Pattuk, Kantarcioglu, Ulusoy, & Malin 2015)).

Feature selection—i.e., choosing at training time a relevant subset of features to include in a model (Guyon & Elisseeff 2003)—can implicitly cut down on the number of features that must be acquired to make a prediction on a new instance at test time. However, typically, the main motivations for feature selection are (1) avoiding overfitting to the training set in high-dimensional problems and (2) generating interpretable models. Some past work (e.g., (Ballesteros & Bolint 2014)) has approached training-time feature selection with the goal of reducing test time prediction costs. Other researchers have pointed out that the optimal budget-constrained set and/or number of features to acquire likely depends on each particular instance. Thus, there is a need for test time, dynamic feature selection.

2.1.1 Cost and order at training time; number at test time

One approach to test time feature selection is learning sequences of features to add. For example, Strubell et al. (2015) learn an ordering of features at training time through the use of prefix scores that capture the prediction margin (i.e., how much more confident the classifier is in the correct prediction than any other). At test time, they then acquire features in this order, computing prefix scores at each stage until some label is predicted above all others by a specified margin. They apply this technique to several problems in NLP (part-of-speech tagging, dependency parsing, and named-entity recognition). A related method is classifier cascades and trees (Xu, Kusner, Weinberger, Chen, & Chapelec 2014), which learn at training time
cost-sensitive trees or cascades, where each node is a classifier. At test time each instance is passed through the tree or cascade, evaluating additional features as necessitated by the tree structure.

While these approaches dynamically decide how many features to acquire in an instance-specific fashion at test time, they do not determine an instance-specific order in which features are acquired at test time. As a result, such methods may have to obtain more features than would be necessary to adequately predict any given instance, just to get the relevant features for that particular instance.

### 2.1.2 Cost at training time; number and order at test time

A more flexible approach is to decide both order and number of features to acquire at test time. Across several domains, test time feature selection has been modeled as a Markov decision process (MDP) (e.g., (He, Daume III, & Eisner, 2012; He et al., 2013; Samadi, Talukdar, Veloso, & Mitchell, 2015; Shi, Steinhardt, & Liang, 2015; Weiss & Taskar, 2013)). Generally, these methods consider the set of features acquired thus far to be the states of the MDP, the decision of which feature to acquire next as the action, with a reward function that reflects how much the inclusion of the next feature improves the prediction (potentially with a penalty on feature cost). They then learn a policy that chooses which action to take (i.e., feature to add) based on the current state (i.e., current known set of features), from a set of training data (e.g., (He et al., 2012, 2013; Samadi et al., 2015; Weiss & Taskar, 2013)).

This is a fairly general approach that has been used quite widely. For example, Shi et al. (2015) take a similar approach to the problem of constructing heterogeneous sampling algorithms, by considering reward as the improvement in conditional log-likelihood of the label given the sample. Similarly, in the domain of recommender systems, the so-called “cold start problem” (Lika, Kolomvatsos, & Hadjiefthymiades, 2014) is improved by eliciting the most relevant preferences from the user to minimize burden (e.g., (Golbandi, Koren, & Lempel, 2011; Sun et al., 2013)). This is often done by learning a decision tree from training data that can be used at prediction time to decide what sequence of ratings to request (e.g., (Golbandi et al., 2011)). A limitation of such approaches is that they use training data to determine how to ask questions (even if the sequence of features depends on previously provided answers); this approach can be inappropriate for a test sample with behavior very different from the training set.

### 2.1.3 Cost, number, and order at test time

Finally, there are methods that determine a feature order at test time, using the expected quality of the subsequent prediction to decide which feature to acquire next. For example, Pattuk et al. (2015) formulate a privacy-aware dynamic feature selection algorithm for classification that sequentially chooses features for a test instance, according to which will most increase the expected confidence of the next prediction, as long as including that feature does not violate a privacy constraint. This work is most responsive to the test time situation. However, it does not address regression, and because its cost metric is defined mathematically by the currently-known features (i.e., conditional entropy), the method cannot take context-dependent costs into account.

### 2.1.4 Summary of key attributes

Table 2.1 summarizes these related works and highlights desired qualities in a test time feature acquisition algorithm. A full consideration of the issues would accommodate a variety of prediction algorithms.

Table 2.1’s Column 1 (Prediction Problem) shows that algorithms may predict discrete values (classification) or continuous values (regression); past work has focused on classification alone. In the Cost Metric column, we see that algorithms may assume that all features have equal cost, or they may allow different features to have different costs, which we refer to as feature-specific costs. Next, algorithms vary on what is done at Test Time rather than training time. The optimal number of features needed, the order of features, and the cost of features may all be determined at test time. If cost is determined at test time, algorithms may be able to consider context-dependent costs. For example, in a system that makes medical diagnoses and can request tests (e.g., (Ferrucci, Levas, Bagchi, Gondek, & Mueller, 2013)), it will be less costly to request an invasive biopsy if a surgery is already scheduled in that area. None of the related works we identified support context-dependent cost metrics. The Utility Function used to determine the value...
of a feature also varies. Examples for feature “quality” include subsequent prediction accuracy (e.g., (He et al., 2013; Shi et al., 2015; Weiss & Taskar, 2013; Xu et al., 2014)) and subsequent prediction uncertainty (e.g., (Pattuk et al., 2015)). Finally, the Domain column shows where each relevant approach was applied.

2.2 Adaptive testing

For tests that measure ability or aptitude, adaptive testing selects test questions based on the respondent’s answer to previous questions. The goal is to measure the examinee’s achievement accurately, without making the examinee answer too many questions. Adaptive tests have been shown to be as reliable and valid as conventional tests (with static question orders), while reducing test length up to 50% (Weiss, 1982). Unlike classical test theory, which assumes all questions equally indicate an assessment outcome, item response theory (IRT) (Lord, 1980) considers individual test questions through an item response function, the probability of a correct answer by an individual at a particular skill level $\theta$. The item response function has three parameters: the pseudo-chance score level (how easy it is to guess the correct answer), item difficulty (how hard it is to answer the question), and discriminating power (how much the skill level influences question response). According to Weiss (1982), an IRT-based adaptive testing framework has the following three components: (1) a way to choose the first item to ask, (2) a way to score items and choose the next item to ask during test administration, and (3) a way to choose to end the test, based on an individual’s performance.

Weiss and Kingsbury (1984) introduce adaptive mastery testing to assess a student’s achievement level $\hat{\theta}$, specifically how the estimated achievement level compares to a “mastery level,” $\theta_m$. At each time point, a question is selected which gives the maximum information at the student’s current estimated mastery level and asked. As the student answers questions, the estimate $\hat{\theta}$ is updated, along with a confidence interval. Once the confidence interval for $\hat{\theta}$ no longer includes $\theta_m$, the test is finished and the student’s mastery level
is assigned as sufficient or not (depending if $\theta_m$ lies above or below the confidence interval for $\hat{\theta}$).

More recently, IRT-based adaptive testing has been used for diagnoses of mental health disorders through patient questionnaires (Gibbons, Weiss, Frank, & Kuper 2016). Their experiments show that their adaptive diagnosis process can, in only one minute of testing, arrive at the same diagnosis as a trained clinician in one hour. Montgomery and Cutler (2013) have also used IRT-based adaptive testing, but for public opinion surveys. In an empirical study using adaptive testing to measure respondents’ political knowledge, the authors found that the adaptive testing approach could produce more accurate measurements than traditional test administration, at a 40% reduction in questionnaire length.

2.3 Surveys

Unlike test-time feature acquisition, the motivation of which is gathering the most relevant information for a prediction, surveys typically have the goal of collecting complete data from a population.

2.3.1 Burden in surveys

Respondent burden in surveys is multifaceted, with multiple factors influencing what is ultimately a subjective measure on the respondent; some of these factors are survey length (e.g., number of questions or estimated time to complete), respondent effort, respondent stress, or frequency of interviews (Bradburn 1978). A respondent’s perception of the survey task has a significant direct effect on self-reports of survey burden (Fricker, Yan, & Tsai 2014). Recent work has demonstrated that telling respondents that they have been screened into a longer or shorter survey, with a longer or shorter expected time commitment, can influence their perception of survey burden, with those told they were screened into a longer survey reporting more burden than those who also received the longer survey but were not informed of their selection (Yu, Fricker, & Kopp 2015).

2.3.2 Motivating and engaging survey respondents

As survey response rates have been dropping (e.g., Porter 2004; Shih & Fan 2008), researchers have been looking at how to motivate respondents to fill out these surveys, to avoid having survey results not represent the full population. Commercial surveys often pay respondents, but compensation does not necessarily ensure thoughtful responses—participants still exhibit satisficing behavior in paid surveys (e.g., Barge & Gehlbach 2012; Kapelner & Chandler 2010). Incentivizing respondents with something dependent on the quality of their answers, like a personalized prediction or calculation, can motivate them to provide data that accurately reflect their situations. For example, Angelovska and Mavrikiou (2013) design an online questionnaire that gives the respondent feedback on their level of procrastination, based on their responses. Their experiments find that the questionnaire that promises feedback has lower dropout rates than the standard questionnaire in which the respondent does not receive personalized feedback.

Another promising venue for increasing respondent motivation is by using paradata collected as the respondent answers an online survey (e.g., time spent on page, mouse clicks (Kaczmarek 2008)) to model user engagement, with subsequent survey action taken to increase user engagement and response rates (M. P. Couper et al. 2010).

2.3.3 Adaptive survey design

Adaptive survey design (ASD) attempts to improve survey quality (in terms of achieving a higher response rate or lower error) by giving respondents custom survey designs, rather than the same one (Schouten, Calinescu, & Luiten 2013). Usually ASD tries to minimize nonresponse, and designs involve factors like number of follow-ups, which can be costly. The general technique is to maximize survey quality while keeping costs below a budget.

Often in ASD, changes in survey design happen between phases of the survey, where the exact same survey protocol (e.g., sampling frame, survey mode, measurement conditions) is in place within a phase and results from that phase inform changes to the protocol for the next phase. Groves and Heeringa (2006) introduce an approach they call responsive survey design, which uses indicators of the cost and error of
design features to make decisions about how to change the survey design in future phases and then combines data from all phases into a final estimator. They also introduce the concept of phase capacity—once a stable estimate has been reached in a design phase, it is unlikely that expending more effort in that phase will result in a better estimate. Their definition of “effort” focuses on collecting participants for each phase. They propose the use of error-sensitive indicators to identify when a phase has reached capacity and no more participants need to be recruited for that phase. This notion of phase capacity could extend to reaching a stable estimate of a participant’s survey-answering, and no more questions need to be asked.
CHAPTER 2. RELATED WORK
Chapter 3

A general framework for question ordering

Our goal is to personalize question ordering for an individual providing information, depending on our current knowledge of them. This knowledge could come from answers they have given to previous questions, as well as information collected through paradata during the information-providing process. Because we do not know ahead of time the budget for question asking, we present a greedy approach for iterative question selection: at each time step, choose the question that optimizes a selection criterion; ask that question and receive an answer; update the current knowledge base; and continue the process until all questions have been asked, the allowable budget for asking questions is exhausted, or the respondent stops providing answers.

3.1 A criterion for iterative question selection

The question selection rule trades off the expected utility of having an answer to a question (or a set of questions) with the cost of getting the answer:

$$q^* = \arg \min_q (-E[U(q)] + \lambda c_q),$$

(3.1)

where $q$ is a question or set of questions, $U(q)$ is the utility of $q$, $c_q$ is the cost of $q$ (which may be context-dependent and therefore not known until evaluation time), and $\lambda$ is a tradeoff parameter. We want to maximize utility while minimizing costs. Figure 3.1 summarizes how this iterative question selection process works for a sample at test time. Definitions of question utility and cost will vary according to the application and the purpose of the data collection; here we present an overview of possible interpretations for utility and cost, which will be expanded for example applications in our experiments in later sections.

This method can be generalized to order modules of related questions, rather than individual questions. Reasons to present questions in modules rather than purely sequentially include (1) presenting related items in a group can reduce the cognitive burden required of a respondent to answer the group (e.g., if a set of questions asks the respondent about various aspects of their commute, as the American Community Survey does, it will be easier for the respondent to answer those commute-related questions as a unit rather than scattered throughout the entire questionnaire) (Tourangeau, 1984) and (2) imposing a standard order on certain questions that are susceptible to order effects (Sudman, Bradburn, & Schwarz, 1996) can ensure that all participants understand and answer questions in the same way, even when question order is determined dynamically.

3.1.1 Question utility

Intuitively, question utility captures how “useful” a candidate question is. The exact definition for utility depends on the application (i.e., what do we value as useful?) and the data (i.e., how can we calculate this value?). For example, if the application is a prediction task, we can use the impact a question will
CHAPTER 3. A GENERAL FRAMEWORK FOR QUESTION ORDERING

Figure 3.1: Our method iteratively increases the set of known features. **Step 1:** Given a set of known features, it first calculates the expected prediction value and cost of acquiring each unknown feature. This is repeated for all unknown features. **Step 2:** It optimizes for the best combination of prediction value (as calculated in Steps 1 and 2) and cost. **Step 3:** The best next feature is acquired. The process can be repeated until all questions have been asked or the budget for information collection has been exhausted.

have on the prediction quality as a definition for utility; one aspect of prediction quality is *certainty*. The calculation of prediction certainty as a measure of utility will depend on the data and the predictive model being used: if the value to be predicted is continuous, one measure for prediction certainty is the width of the prediction interval, where a wider interval means a less certain prediction; the mathematical definition for the prediction interval width will depend on the predictive model being used. If the value to be predicted is discrete, one measure for prediction certainty is the distance of the sample from the decision boundary; again, calculation for this measure of certainty (the distance from the decision boundary) will depend on the predictive model.

To be dynamic, these definitions of utility will take into account information already known about the sample. Such information may come from previously answered questions or paradata collected in the survey process.

This thesis illustrates examples of the following utility functions: prediction certainty (for continuous outputs, in Sections 4.4.2 and 4.4.3, and discrete outputs in Section 4.4.4), total response variation (Section 4.5), and information gain (Sections 5.3.1 and 5.3.2).

3.1.2 Question cost

The cost of a question reflects how “difficult” it is to get an answer for that question. Different applications have different cost measures that are best suited to them. Examples of cost include (1) the amount of resources needed to answer the question (e.g., time, money, battery, effort), (2) the likelihood that the question will not be answered (i.e., item nonresponse rate), (3) the likelihood that the question will cause the respondent to stop the survey (i.e., item breakoff rate), and (4) a combination of multiple types of cost. These costs can be predefined according to rules or determined empirically from collected data.

When determining question order is deferred to test time, these costs can be *context-dependent*; that is, the current situation of data collection can inform the costs of future questions that might be asked. For example, in a system that makes medical diagnoses and can request tests (e.g., [Ferrucci et al., 2013]), it will be less costly to request an invasive biopsy if a surgery is already scheduled in that area.

This thesis illustrates examples of the following costs: effort required to answer questions (Section 4.4.2), battery drain of collecting mobile data (Section 4.4.3), time to compute features (Section 4.4.4), item response times (Section 4.5), and item breakoff rates (Sections 5.3.1 and 5.3.2).

3.2 Assumptions and requirements

Most generally, this approach requires only definitions of question utility and cost and ways to calculate the expected utility of each candidate question given what is known. In practice, a training set of question responses is needed: measures of utility often depend on statistical properties of samples and estimators, and distributions of question responses need to be known to calculate expected values.
Chapter 4

Prediction-guided question ordering

Here we consider the scenario in which a user is providing information to receive a personalized prediction based on the information they provide. We assume a predictive model has already been developed from a training set, and we want to make a prediction on a new test point. In this setting, it is likely that not all features will be needed to make a reasonable prediction; therefore, we want to ask the most cost-effective set of questions that will maximize prediction quality while minimizing collection costs. Additionally, in this scenario, users receive a direct benefit (i.e., the personalized prediction) and are inherently motivated to provide information. On the other hand, feedback in the form of sequential predictions can influence their decision to continue providing information, if they feel that they have received a useful enough prediction.

When features are being collected to refine a prediction, a natural utility metric to use for test-time feature selection (Equation 3.1) is one that reflects how an additional feature \( f \) contributes to the quality of the subsequent prediction. An ideal solution would accommodate a variety of prediction algorithms, allow for feature-specific and context-dependent costs, and support multiple options for prediction utility when requesting information to refine a prediction. Delaying the order determination until test time (rather than learning it from training data) is a requirement for context-dependent costs (since they are not known until test time).

We developed an approach to test-time Feature Ordering with Cost and Uncertainty Score (FOCUS) that has all of these key properties. A basic assumption of our approach is that it is being applied in the context of supervised machine learning. In addition, we assume that feature cost is only an issue at test (or more generally deployment) time—at training time, the complete set of features associated with each label is assumed to be available. Finally, although not required, our approach benefits from a prediction algorithm that is robust to making predictions when not all features are available (predictions on partial information).

FOCUS is an instantiation of the question ordering framework from Chapter 3 where the utility of a feature is defined to be the certainty of the subsequent prediction made using all known features plus the newly acquired feature. We chose to optimize for prediction certainty due to past work demonstrating the importance of giving people estimates of certainty along with predictions (e.g., (Hirschberg et al., 2011)). Prediction certainty is available in most prediction algorithms and reflects how likely the true value is to coincide with the estimated value. For example, in regression, a prediction interval width indicates the range of values the true value is likely to fall within (Weisberg, 2014). A narrower prediction interval corresponds to a more certain prediction. For classification, certainty can often be formulated as distance from the decision boundary.

As shown in Figure 3.1, FOCUS operates iteratively at test time. Given an instance with known (dark green) and unknown (white) features, it first calculates the expected value of a feature (Step 1) for all features that are not known. Next it calculates the best next feature by optimizing a function that combines the prediction value of each feature with its cost (Step 2). The new feature is acquired (Step 3) and the process repeats. At any iteration, a prediction can be made.

Whether or not the prediction at each step is shown to the user will depend on the application. For example, if the application is gathering information from sensors with no user input, it does not make sense to display the sequential predictions to the user. In this case the algorithm may stop when costs become too high, all features are acquired, or accuracy achieves a certain threshold. This can be determined on an
application-by-application basis. On the other hand, if the user is answering questions to get a personalized prediction, then showing them the partial predictions can keep them engaged in answering questions and help them to decide if continuing to answer questions is valuable to their goals.

4.1 The FOCUS algorithm

FOCUS assumes that a model trained on all feature values is available and that, for a new test point, we want to provide the best (and lowest cost) prediction possible, given that feature values are costly to acquire. As shown in Figure 3.1, FOCUS sequentially estimates the value of each feature (Step 1) and selects a next feature to ask (Steps 2 and 3).

4.1.1 Calculating the utility of a feature $f$

In Step 1 of FOCUS, the expected utility of a feature is calculated. Since the true next prediction uncertainty depends on the actual value for the next feature that is acquired, we cannot directly calculate $E[U(f)]$. However, we can break this down into two parts.

Calculating $U(f)$, the prediction utility for a specific possible value of feature $f$, depends on the exact nature of the prediction problem. $U(f)$ takes as input a feature vector containing the known features plus a hypothetical value $r$ for feature $f$. Typically, utility is calculated by making a partial prediction using those values and estimating prediction accuracy, uncertainty, etc.

This is repeated for all values $r$ that are in the range of potential values $R$ for $f$ (Step 1 of Figure 3.1). If a feature is continuous, we pick bins appropriate for the values that appear in the training set, and then the midpoint for each bin for feature $f$ is used as the set of values $f$ can take on.

There have been several approaches to making predictions under partial information, and FOCUS is compatible with any of them. Reduced-feature models use only features whose values are known in making predictions; these models may be calculated at training time (e.g., (Xu et al., 2014)) or dynamically constructed at test time (e.g., (Friedman, Kohavi, & Yun, 1996)). Another option is to impute missing values and use the full-feature model on the combination of known and estimated features to make a prediction. Hybrid approaches combine reduced-feature modeling with imputation (e.g., (Saar-Tsechansky & Provost, 2007)). However, in the end, FOCUS is agnostic about how predictions are made and how $U$ is defined.

4.1.2 Calculating the expected prediction utility of a feature $f$

Given a way to calculate $U(f)$, $E[U(f)]$ can easily be defined. We calculate the expected utility of a prediction that includes feature $f$ by taking a weighted average of the utility calculated for each possible value of $f$:

$$E[U(f)] = \sum_{r \in R} p(z_f = r) U(z_{f:=r}),$$

where $p(z_f = r)$, the probability that the $f$-th feature’s value is $r$, is calculated empirically from the training set, and the notation $z_{f:=r}$ means that the $f$-th component of feature vector $z$ is replaced with the value $r$.

Algorithm 2 summarizes this process in pseudocode.

This process is then repeated for all unknown features.

4.1.3 Optimizing for the best next feature

In its middle step (for each iteration), FOCUS optimizes for utility of the next feature, penalized by the cost of that feature. Our selection rule, illustrated in Step 2 of Figure 3.1, trades off the expected utility of the next prediction, for each candidate feature, with the cost of that feature:

$$f^* = \arg \min_{f \in K} (-E[U(f)] + \lambda c_f),$$

where $E[U(f)]$ is the expected utility calculated in Step 1 of FOCUS, and $\lambda$ controls how much weight we give to the cost $c$ for each feature. Algorithm 3 summarizes this process in pseudocode.
As with utility, cost is calculated in a problem-specific fashion, based on the feature vector of currently known features. This allows the cost function to consider context-dependent information such as the values of other features that are already known. Cost could be measured in time necessary to acquire a feature, either computationally or due to dependencies; direct impact on the user such as interrupting her to ask a question; or indirect impact on the user such as drawing down the battery life of her phone.

4.2 Regression: Predicting continuous values

Our first two applications (predicting household energy consumption and predicting student stress levels) involve predicting continuous values, so we use prediction interval width to measure uncertainty: a narrower prediction interval corresponds to a more certain prediction. Because we are imputing missing features to make predictions with partial information, we use the measurement error model (MEM) [Fuller 2009] to capture error associated with estimated features. Unlike traditional regression models, MEMs do not assume we observe each component \( x_f \) exactly; there is an error \( \delta_f \) associated with the estimation:

\[
z_f = x_f + \delta_f, \text{ where } \mathbb{E}[\delta_f|x_f] = 0.
\]

Prediction \( \hat{y} \) still depends on the true, unobserved value \( x \):

\[
\hat{y} = \hat{\beta}^T \hat{x} = \hat{\beta}(\bar{z} - \bar{\delta}),
\]

where \( \hat{\beta} \in \mathbb{R}^{d+1} \) is the parameter vector learned on the training set \( X \) (recall that all feature values are known at training time). The notation \( \bar{x}, \bar{z}, \bar{\delta} \) means vectors \( x, z \) have a 1 appended to them and \( \delta \) a 0 to account for the constant term in the regression. Let \( \bar{X} \) extend this notion to the training matrix: \( \bar{X} = [1^n X] \).

We can calculate a 100(1 - \( \alpha \))% prediction interval for a new point \( z \) as

\[
\hat{y} \pm t_{n-d-1, \alpha/2} \sqrt{\hat{\sigma}^2 (1 + \bar{z}^T(\bar{X}^T\bar{X})^{-1}\bar{z} + \bar{\delta}^T(\bar{X}^T\bar{X})^{-1}\bar{\delta})},
\]

(4.3)

where the \( \bar{\delta}^T(\bar{X}^T\bar{X})^{-1}\bar{\delta} \) term accounts for error from estimated features and \( t_{n-d-1, \alpha/2} \) is the value at which a Student’s \( t \) distribution with \( n - d - 1 \) degrees of freedom has cumulative distribution function value \( \alpha/2 \). We can estimate \( \delta \) from training data by calculating the error of predicting each feature with kNN, from the other features. We also estimate \( \hat{\sigma}^2 \), the regression variance, from training data.

When using prediction certainty as the measure of feature utility (where feature utility means how much a feature will influence the prediction quality), a more certain prediction (i.e., a more useful feature) will have a smaller prediction interval width than a less certain prediction. Thus, in this case, we want to minimize prediction interval width (equivalent to minimizing uncertainty, or maximizing certainty/utility), so we use the negative prediction interval width as the utility measure for Equation 3.1, which then reduces to

\[
f^* = \arg \min_f (\mathbb{E}[W(f)] + \lambda c_f),
\]

(4.4)

where \( \mathbb{E}[W(f)] \) is the expected prediction interval width of the next prediction that includes feature \( f \).

4.3 Classification: Predicting discrete values

The utility metric used to measure prediction uncertainty for regression, the prediction interval width, is not applicable for classification. However, we can define a measure of prediction uncertainty that is appropriate for classification. For classification, the probability that a test instance belongs to a particular class is the certainty of its prediction—a lower class probability—means a less certain prediction. Thus, in this case, we want to minimize prediction interval width (equivalent to minimizing uncertainty, or maximizing certainty/utility), so we use the negative prediction interval width as the utility measure for Equation 3.1, which then reduces to

\[
f^* = \arg \min_f (\mathbb{E}[W(f)] + \lambda c_f),
\]

(4.4)

where \( \mathbb{E}[W(f)] \) is the expected prediction interval width of the next prediction that includes feature \( f \).
4.4 Applications and experiments

To demonstrate the usefulness of FOCUS for cost-effective interactive predictions, we implemented it for several prediction algorithms and applications. First, we consider the case of providing personalized energy estimates for prospective tenants, where feature cost reflects how much effort a user must exert to provide answers about their energy-consuming habits and their new potential home. Next, we use FOCUS to make momentary predictions of stress in college students, where feature cost includes both battery drain from turning on mobile sensors in addition to the cost of interrupting the user to ask for feature values. In this example, we also consider context-dependent costs, in particular, the fact that the cost turning on a sensor at the expense of draining the battery is no longer an issue when the phone is charging. Finally, we apply FOCUS to the classification problem of identifying devices in photos to support opportunistic interactions with low user burden.

For each of these applications, we compare the quality of successive predictions obtained with our method, FOCUS, with a variety of cost penalties $\lambda$, to that of a fixed-order baseline, which we call Fixed Selection. This baseline acquires features in the order of forward selection (Harrell, 2001; Tropp, 2004) on the training data (resulting in an identical ordering for all samples). Finally, we implement an Oracle that chooses the next best feature to acquire according to the minimum true utility of the next prediction, rather than the expected utility, as in Equation 3.1.

In all three applications, we use imputation to predict with partial features as follows: Using $k$NN (Cover & Hart, 1967), restricted to the features that are already known, we find the $k$ points in the training set that are nearest to the test point. We estimate the value for each unknown feature in the test point as the mean or mode of that feature in the $k$ nearest neighbors (see Algorithm 1).

4.4.1 Metrics for prediction quality

Our validation considers prediction certainty, error, and cost. Certainty and cost are defined as part of our optimization problem. For error, we use mean absolute error for regression and zero-one loss for classification (i.e., a sample incurs an error of 0 if its predicted value matches the true value and 1 otherwise) to compare our successive predictions to the true values. This is a conservative metric for error since it compares only a single predicted value to the true value (rather than taking into account the uncertainty associated with the prediction); for example, this metric will incur error when the true value is not the exact midpoint of a prediction interval, even when the true value does lie within the prediction interval.

Figure 4.1 illustrates the cost savings of FOCUS (solid lines) over the baseline (dashed lines) when various numbers of additional features have been provided for each of our three validation applications. The left plot (Figure 4.1a) shows that, as the cost penalty $\lambda$ increases, the cost savings of FOCUS over the baseline also increases. As expected, increasing the cost tradeoff parameter $\lambda$ favors asking inexpensive features near the beginning of the test time feature acquisition process. The right plot (Figure 4.1b) shows that, for a fixed $\lambda$ and with increasing numbers of additional features, FOCUS also maintains its cost advantage over the baselines, for all three applications.

To summarize the trajectories of prediction cost, uncertainty, and error, we calculate areas under the curve for each of the metrics as each feature is added. Smaller values are better because they mean the algorithm spent less time in high cost, uncertainty, and error. Table 4.1 lists these values for FOCUS with seven different values of cost penalty $\lambda$, ranging from zero to one, and for the baseline. As expected, due to the two terms in the selection rule (uncertainty and cost), cost decreases as the penalty on cost increases, and uncertainty tends to increase as the cost penalty increases. There is no pattern in how error changes as $\lambda$ increases. The baseline (Fixed Selection) error is often lower than FOCUS; this result is not surprising, due to the error-minimizing criterion of forward selection. FOCUS with a nonzero $\lambda$ always has significantly lower cost than the baseline. Prediction uncertainty is sometimes lower with FOCUS than baseline (particularly for low values of $\lambda$). The metric that suffers the most with FOCUS, relative to the baseline, is error because it was not included in the optimization.
4.4. APPLICATIONS AND EXPERIMENTS

Figure 4.1: Charts showing impact on cost (y axis) of λ and number of features, for FOCUS (solid lines) and the Fixed Selection baseline (dashed lines).

(a) As the cost penalty \( \lambda \) increases, the cost savings from FOCUS increase, without significant loss of accuracy, compared to the fixed order baseline. On this plot, the number of features \( n \) is fixed at 4.

(b) FOCUS ensures that cost increases more slowly than the baseline, with \( \lambda = .001 \). The Student Life and Device ID sets converge because there are only 8 and 12 total features; RECS converges at 30 features (not shown).

4.4.2 Predicting energy usage for prospective tenants

Selecting energy-efficient homes is important for renters, because in many climates energy costs can be a significant burden, and the choice of infrastructure influences energy consumption far more than in-home behavior (Dietz, Gardner, Gilligan, Stern, & Vandenbergh, 2009). However, there is a paucity of information available about expected energy costs pre-lease signing. Calling a utility to ask about prior costs may give incomplete or misleading information (since occupant behavior can influence energy usage as much as 100% (Seryak & Kissock, 2003)), and a carbon calculator typically requires users to answer (prohibitively) many questions that may require considerable research to answer. We can lower user burden by (1) learning the relationship between household features (home infrastructure and occupant behavior) and energy use from established datasets, such as the Residential Energy Consumption Survey (RECS) (U.S. Energy Information Administration, 2009), and then (2) using FOCUS to strategically select which instance-specific features are needed to make a confident prediction for a new household.

Background

Bottom-up methods for modeling residential energy consumption use features of individual households (Swan & Ugursal, 2009). Household features may include macroeconomic indicators, as well as occupant-specific features (e.g., Douthitt (1989), Kaza (2010)). For example, Douthitt (1989) examines fuel consumption for space heating in Canada, using household-specific values for occupant demographics, the housing structure, and fuel cost. Kaza (2010) uses the Residential Energy Consumption Survey (RECS) to estimate energy usage from low-level housing characteristics, with a quantile regression approach to separate the effects of variables on homes with different patterns of energy consumption.

The RECS dataset is our focus as well. RECS consists of data from 12,000 households across the United States, with energy consumption by fuel type (e.g., electricity, natural gas) and around 500 features of each home and its occupants. We restrict our use of RECS to electricity prediction in a single climate zone where energy consumption is variable due to cold weather, giving us a subset of 2470 homes.

Cost metric

In Early, Mankoff, and Fienberg (2016), we assign cost using a three point scale: some features are free (available in rental advertisements), while the remainder are either low cost (e.g., number of occupants)
or high cost (e.g., age of heating equipment). Free features are included in all predictions since they have no cost. Here, we use a more nuanced eight-point feature cost scale based on difficulty of acquiring a feature. Zero-cost features are “free” (i.e., extractable from a rental listing); 1-2 are occupant-related; and 3-7 are unit-related (may require a site visit and/or research). Table 4.2 lists the cost categories and an example feature in each. Table 4.3 justifies our choice of “free” features (i.e., features extractable from rental advertisements) by showing how frequently those features appear in the database of Rent Jungle, an online rental search platform.

Table 4.2: Feature cost reflects how difficult it is to acquire a value for a feature. Note that the cost of a feature might vary by home: for example, if a listing included pictures of the kitchen, a user could see the door arrangement of the refrigerator from the listing (cost 3). If there were no pictures, then it would be an easily-visible feature (cost 5).

Table 4.3: Areas under the curve for the cost, uncertainty, and error metrics from FOCUS, with a variety of cost penalties $\lambda$, and the Baseline (Fixed Selection): smaller values mean the algorithm spent less time in high uncertainty, error, and cost. Ideally, FOCUS will have lower cost and similar uncertainty as Baseline. This is shown in black. Numbers marked with * show where FOCUS was significantly lower than baseline at the $\alpha = 0.05$ level. Numbers in red, marked with †, indicate where FOCUS did significantly worse than baseline (higher cost, uncertainty, or error) at the $\alpha = 0.05$ level.

<table>
<thead>
<tr>
<th>Method</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RECS</td>
<td>StudentLife</td>
<td>Device ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1340.25</td>
<td>41.376*</td>
<td>12.457</td>
<td>71.182*</td>
<td>22.897</td>
<td>5.770</td>
<td>94.788*</td>
<td>1.935</td>
<td>4.815</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1175.42</td>
<td>41.375*</td>
<td>12.444</td>
<td>70.924*</td>
<td>22.898</td>
<td>5.769</td>
<td>87.647*</td>
<td>1.939†</td>
<td>5.241</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.0001</td>
<td>936.338*</td>
<td>41.371*</td>
<td>12.649†</td>
<td>70.030*</td>
<td>22.898</td>
<td>5.771</td>
<td>87.601*</td>
<td>1.940†</td>
<td>5.259</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.01</td>
<td>891.130*</td>
<td>41.430*</td>
<td>12.578†</td>
<td>66.704*</td>
<td>22.899</td>
<td>5.775†</td>
<td>87.638*</td>
<td>1.936</td>
<td>5.296†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.1</td>
<td>885.000*</td>
<td>41.446</td>
<td>12.704†</td>
<td>63.303*</td>
<td>22.929</td>
<td>5.795†</td>
<td>82.001*</td>
<td>1.936</td>
<td>5.463†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>885.000*</td>
<td>41.446</td>
<td>12.704†</td>
<td>61.303*</td>
<td>22.958</td>
<td>5.800†</td>
<td>80.547*</td>
<td>1.936</td>
<td>5.574†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baseline</td>
<td>1250.000</td>
<td>41.447</td>
<td>11.722</td>
<td>81.000</td>
<td>22.898</td>
<td>5.705</td>
<td>105.000</td>
<td>1.928</td>
<td>4.796</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Experimental setup

We first divide RECS into training (90%) and testing (10%) sets. At training time, we use forward selection ([Tropp 2004]) on a randomly-selected subset of 20% of the training data to choose 30 higher-cost features to add to the free features for prediction. We use ten-fold cross validation on the remainder of the training data to learn regression weights.

---

http://www.rentjungle.com
Table 4.3: The features we define as extractable (i.e., “free”) appear in most of the listings on Rent Jungle. Geographic features associated with the city, zip code, or state include climate zone and whether the area is urban or rural, among others.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Presence in Rent Jungle database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of bedrooms</td>
<td>85%</td>
</tr>
<tr>
<td>Number of full bathrooms</td>
<td>57%</td>
</tr>
<tr>
<td>Studio apartment</td>
<td>85%</td>
</tr>
<tr>
<td>City or zip code</td>
<td>99%</td>
</tr>
<tr>
<td>State</td>
<td>100%</td>
</tr>
</tbody>
</table>

At test time, the goal is to make a prediction on a new test point and acquire costly features as needed to improve the prediction. We simulate progressive addition of features by hiding values for “unknown” features, using FOCUS to choose a feature to acquire at each step, and unveiling that feature’s value once it is “asked” and “answered.”

Results

The regression model using FOCUS had significantly lower costs than the baseline for all values of except 0 with equivalent or better uncertainty (Table 4.1). For \( \lambda = .001 \), FOCUS yielded an average of 45% savings in cost compared to the baseline as features were added to the prediction. As expected, FOCUS performed worse in terms of error, since FOCUS optimizes prediction uncertainty, rather than error, when determining a test-specific feature order; the baseline was chosen by optimizing prediction error on the training set. However, for this application in particular, it is more important that predicted energy costs fall inside the window of uncertainty (which the definition of a prediction interval ensures) than that they have an accurate dollar value (Hirschberg et al., 2011).

4.4.3 Predicting student stress levels

Knowing a user’s stress level at an upcoming point in time allows for automatic suggestions or reminders to help them manage stressful situations. We want to predict momentary stress reports from college students, given an assortment of data such as demographic information, depression, sleep habits, and deadlines (Wang et al., 2014). This task has redundant features with various costs. For example, we can ask a student to fill out a survey to measure their anxiety, or we can use phone sensors to measure length of sleep; the first method incurs the cost of interrupting the person (potentially at a bad time), the second the cost of draining the phone battery. Our goal is to arrive at a “good” (low-uncertainty) prediction for stress levels without exhausting too many resources by strategically selecting which features to obtain.

This application differs from the (simpler) personalized energy problem in the previous section in several key ways: (1) We want to predict momentary stress levels, rather than a single value constant across time, as in the energy prediction example; (2) We include a new type of cost: battery life; (3) We consider context-dependent costs: for example, when a user’s phone is charging, turning on sensors is no longer prohibitively expensive.

Background

Biologically meant as a mechanism for survival, stress can become harmful if sustained for long periods of time (Moberg, 2000), with individual-level health and societal-level economic consequences (Kalia, 2002). College students face a unique and significant type of stress, partly because of their transition between dependent child and independent adult (e.g., Ross, Niebling, & Heckert, 1999).

It is possible to estimate stress from physiological and physical signals, like skin conductance, brain activity, and pupil dilation (e.g., Sharma & Gedeon, 2012); however, these measurements often require unwieldy, task-specific instrumentation. However, more accessible signals that could be measured in a
lightweight fashion (from mobile phone data) are also associated with stress, like movement, heart rate, sleep length, and social activity (e.g., (P. Ferreira, Sanches, Höök, & Jaensson 2008; Hudd et al. 2000; Misra & McKean, 2000)).

For example, Affective Health (P. Ferreira et al. 2008) senses bodily reactions (such as movement and heart rate) and visualizes them in real time, giving users the opportunity to connect their activities to their mental state. The StudentLife project (Wang et al. 2014) collected smartphone data from 48 graduate and undergraduate students over the course of an academic term and included self-reported stress, which was correlated with other factors such as GPA. Participants in the Student Life project provided nearly 1600 self-reports of stress levels, with individuals providing between 3 and 269 responses. Figure 4.2 illustrates the number of stress reports and average stress levels for each participant.

Figure 4.2: StudentLife participants’ number of stress reports (y axis) and average stress (indicated by color and marker size).

The StudentLife dataset is our focus as well, but we restrict our analysis to predicting stress. Our goal is to reduce the burden on users of answering experience sampling questions (the current approach to measuring stress in the StudentLife project) by substituting other features when the impact on prediction would be minimal. Thus, we use self-reports of stress at various time points as our response variable.

Cost metrics

The cost of acquiring features depends on battery drain (low for sensors like detecting light or if the phone is charging; high for sensors like the accelerometer and microphone (e.g., (D. Ferreira, Kostakos, & Dey 2015; Lu et al. 2010))) and costly interruption of the user (asking for amount of sleep or upcoming deadlines). Some of these costs are context-dependent: e.g., if a phone is charging, battery drain from turning on mobile sensors is no longer an impediment to gathering those features.

Experimental setup

As in the energy application, we used linear regression to predict stress reports. However, rather than using feature selection to choose a subset of features from a larger set, we used previous research findings to extract features relevant to stress (e.g., (Hudd et al. 2000; Misra & McKean 2000)), along with some current context: time of day, sleep length, exercise length, length of time until next deadline, number of upcoming deadlines, current activity (stationary or in motion), current audio (silent or noisy), if the phone is currently
in a dark environment, and if the phone is currently charging. We assume that time of day is freely available. We assign three additional cost categories: lower-cost sensors (i.e., light detection and charging); higher-cost sensors (i.e., accelerometer to measure activity and microphone to measure audio); and highest-cost user interruption (to ask questions about lengths of sleep and exercise and upcoming deadlines).

We restricted our dataset to users who provided stress, deadline, and exercise information, resulting in a total of 660 individual stress reports. We used 80% of these stress self-reports for training (i.e., learning regression weights) and the remaining 20% for testing with FOCUS.

Results

The regression model confirms several well-known properties of stress. For example, exercise is negatively correlated with stress, and number of deadlines is positively correlated with stress. Table 4.4 summarizes the predictor learned on the training set. As with the prior dataset, our approach results in predictions that are more certain (i.e., have narrower prediction interval widths) than the baseline, Fixed Selection, while being similarly accurate. Stress prediction is shown with red lines in Figure 4.1 (which shows cost improvement of FOCUS over baseline). Accuracy levels did not differ significantly for stress predictions for any of the combinations of values shown in Figure 4.1. Prediction certainty decreased slightly for $\lambda$ after $n = 5$. On average, FOCUS yielded 23% savings in cost compared to the baseline as features were added to the prediction.

Table 4.4: Regression weights from our linear model to predict stress from sensed and user-provided data in StudentLife.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Regression weight</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>-0.5401</td>
<td>0.1471</td>
</tr>
<tr>
<td>Time of day</td>
<td>-0.0295</td>
<td>0.7950</td>
</tr>
<tr>
<td>Sleep</td>
<td>0.6187</td>
<td>0.2281</td>
</tr>
<tr>
<td>Exercise</td>
<td>-0.1151</td>
<td>0.2912</td>
</tr>
<tr>
<td>Time to deadline</td>
<td>-0.0401</td>
<td>0.8559</td>
</tr>
<tr>
<td>Number of deadlines</td>
<td>0.2625</td>
<td>0.1137</td>
</tr>
<tr>
<td>Currently moving?</td>
<td>-0.0226</td>
<td>0.8868</td>
</tr>
<tr>
<td>Currently silent?</td>
<td>0.0479</td>
<td>0.6125</td>
</tr>
<tr>
<td>Currently dark?</td>
<td>-0.0864</td>
<td>0.2117</td>
</tr>
<tr>
<td>Currently charging?</td>
<td>0.0492</td>
<td>0.5649</td>
</tr>
</tbody>
</table>

An important question is whether context-dependent feature costs have an impact on feature ordering. To answer this, we divided the test data into stress reports that were given when the phone was charging (20% of the reports) and those that were given when it was not plugged in. Feature order should differ in those cases, since sensor feature cost is zero when the phone is charging. Figure 4.3 shows how frequently each feature was chosen in each position, for the charging and noncharging contexts, when cost penalty parameter $\lambda$ equals 1. The battery-draining sensors are in the last four columns. When the phone is charging (Figure 4.3a), the sensed features tend to be added before the user-answered features and without regard for the relative cost of acquiring sensed features (top right 4x4 corner). When the phone is not charging (Figure 4.3b), features tend to be asked in order of increasing costmost inexpensive sensed features first (top right 2x2 corner), followed by more expensive sensed features (middle 2x2 section), and finally by the most expensive user-provided features.

4.4.4 Identifying devices in photographs

Correctly identifying a device (e.g., printer, projector) in an image can support opportunistic mobile interaction with that device by automatically installing the necessary drivers without forcing the user into a manual setup. Real-time interaction speeds are crucial in this setting to make the opportunistic interaction seem truly seamless, but often image classification algorithms require computing expensive (i.e., time-consuming)
CHAPTER 4. PREDICTION-GUIDED QUESTION ORDERING

(a) When phone is not charging, sensed features are more likely to be added in order of increasing cost (the low-cost sensor features are added before the high-cost sensor features, shown by the block structure in the bottom right corner).

(b) When the phone is charging, acquiring sensor features is no longer expensive, and so sensed features (last four columns) are requested more uniformly (bottom right corner) since there is no distinction between low- and high-cost sensors.

Figure 4.3: Heatmaps showing how frequently features were chosen in each position, for the two cases for context-dependent costs: when the phone is not charging and when it is charging. Color indicates frequency; x axis indicates feature type; y axis represents feature order.

features that can slow down the classification. The device identification problem can take advantage of other, less time-consuming features, like location and camera orientation to assist in prediction. User input (e.g., desired use of a device, such as printing or projecting) can also inform the prediction, at the cost of user inconvenience and time.

We use a dataset of images, camera orientations, photo locations, and device capabilities (i.e., “can print,” “can scan”, “can copy,” “can fax,” and “can laser-cut,” for this dataset) to classify new images as particular devices (de Freitas et al., 2016). We illustrate the usefulness of FOCUS for this device classification task with decision trees and show that using FOCUS to select a dynamic subset of features to acquire at test time results in fewer expensive feature acquisitions while still correctly classifying devices.

Background

Multiple groups have considered how smartphones can be used to control physical devices, with a variety of device identification and control mechanisms. Examples include laser pointers (e.g., Beigl, 1999), external cameras (e.g., Budde et al., 2013) with Kinect, and magnetometers (e.g., Wu, Pan, Zhang, Li, & Wu, 2010). For the case of smartphone-taken images, past work explores directly identifying appliances labeled with fiducial markers (e.g., Liu, McEvoy, Kimber, Chiu, & Zhou, 2006) or using image recognition to identify a pictured device (e.g., Chang & Li, 2011). Snap-To-It (de Freitas et al., 2016) allows users to interact with new devices by taking a picture with their phones and using both the content and context of the image to identify the pictured device and connect to it.

Snap-To-It is our focus as well, and we also use image content and context (location and camera orientation) to classify devices, as well as user input about intended device use. As in Snap-To-It, we use the Scale-Invariant Feature Transform (SIFT) algorithm to extract features from images and compare SIFT features from two images for matches—a higher number of matches means that the images are more similar (Lowe, 1999). It is possible to compute the SIFT features for the 90 reference images ahead of time, but at prediction time, the SIFT features for the image the user takes must be calculated and then compared to the reference images to check for the highest match. Our experiments show that calculating SIFT matches for a new image, against precomputed SIFT features for all 90 reference images takes, on average, 2.80
seconds, which can destroy the “real-time” feel of a service like Snap-To-It. Asking for user input is also expensive. Therefore, our goal is to give confident device predictions for images with few time-consuming SIFT match operations and overall low inconvenience on the user.

**Cost metrics**

We assume that image location and orientation are freely available when the picture is taken. We assign two additional cost categories: medium (SIFT matching) and high (asking the user about their desired use for the device). Although additional context could be relevant (such as whether PowerPoint is running and a projector is in the room), the Snap-To-It dataset did not include this.

**Experimental setup**

The Snap-To-It dataset is pre-divided into “reference” and “testing” subsets. The reference dataset contains five images for each of 18 appliances, taken from different angles. These appliances have printing, scanning, copying, faxing, and laser-cutting capabilities. There are 108 images (six of each appliance) in the testing set. We used the reference set to construct a decision tree for image classification. Then we used this decision tree to classify test images, computing the SIFT matches and user questioning as determined to be necessary with FOCUS.

**Results**

We first constructed a decision tree on the Snap-To-It reference set, using MATLAB’s implementation of the Classification and Regression Tree (CART) algorithm. CART is a top-down algorithm that repeatedly splits nodes of the tree (starting with all samples at the root), according to whichever binary split most decreases the “mixture” among classes in the leaves, measured by the Gini impurity \(\text{Breiman, Friedman, Stone, \\
\& Osslen [1984]}\). The decision tree learned was able to optimally classify the reference set using only 7 of the 90 reference images, so we discarded the rest.

Device identification performance is shown with blue lines in Figure 4.1 (which shows cost improvement of FOCUS over baseline). On average, FOCUS yielded 29% savings in cost compared to the baseline as features were added to the prediction. In comparison to the baseline, accuracy was significantly worse only at \(n = 10 − 11\) for \(\lambda = .001\).

These experiments illustrate two valuable ways of cutting down time-consuming test time image comparisons. First, using a decision tree to classify instances reduced the potential image comparison space from 90 to 7; de Freitas et al. [2016] used heuristics from image location and orientation to reduce the space of potential matches, but an algorithmic approach can expand the impact of such filtering beyond human-extractable patterns. Second, using test time feature acquisition can further reduce the number of costly feature acquisitions on test instances that can be confidently classified without obtaining all features.

**4.4.5 Discussion of results**

Making real-time, personalized predictions is an important opportunity for ubiquitous computing applications; however, gathering information from users at test time can be costly, especially when not all pieces of information may be relevant for a particular user at a particular time. We have demonstrated the cost-saving value of dynamically acquiring features for test time prediction on a variety of applications and algorithms. On all three validation datasets, FOCUS effectively lowered prediction costs (by reducing the number of additional, costly features to acquire), without sacrificing prediction quality for most values of \(n\) and \(\lambda\). The FOCUS framework’s ability to support context-dependent costs (illustrated in the stress prediction example on the StudentLife dataset) allows for richer, more realistic interpretations of feature cost, which may not be fixed for all test instances.

A limitation of our work is our simplistic measure of costs for all of our predictions. A more detailed look at cost could account for real users’ perception of question cost (estimated via item response times or response rates) or the exact battery drain of various sensors on the particular model of phone being used. Furthermore, future work should explore how to connect the end-user experience to choosing a value for the cost penalty \(\lambda\); this tradeoff is likely application-specific.
Additionally, in the stress prediction example, we considered battery charging state (i.e., whether or not the phone was currently charging) as a simple binary influencer for context-dependent costs (with cost set to 0 when the phone was charging). However, more nuanced contexts could take into account the current percentage of remaining battery power, the current drain on the battery based on what applications are currently running (e.g., (D. Ferreira, Ferreira, Goncalves, Kostakos, & Dey, 2013; Min et al., 2015)), or the expected time-to-next-charge (e.g., (Banerjee, Rahmati, Corner, Rollins, & Zhong, 2007; Ravi, Scott, Han, & Ifode, 2008)). It would also make sense for this application to consider the influence of user context on the cost of asking them a question—e.g., if a user’s calendar indicates they are currently in a meeting, it may not be a good time to acquire a feature that requires user input.

Similarly, some cost metrics might take into account whether features are “shared” for multiple needs—e.g., if someone answers a stress EMA multiple times in one day, the “day-level” features can be shared across predictions.

Another aspect worth considering in test time feature acquisition is feature confidence, especially when the same value can be obtained through different methods, with different costs and accuracies. For example, in the StudentLife case we used user-provided sleep lengths as one of the predictors for stress, but it is also possible to estimate sleep length and quality by sensing. By incorporating feature confidence into the selection criterion, we could decide whether we are confident “enough” about a value for a less costly feature to avoid acquiring a more expensive estimation of the same value.

Finally, it might be interesting to explore user- and context-specific metrics for prediction quality. Thus, the current needs of a user (in terms of accuracy) might be factored in to choices about which features are worth acquiring.

### 4.5 Questionnaire scoring

A popular use of questionnaires is to summarize an individual’s responses into a single score, for a variety of applications, such as psychology, education, and health. Often, this single score measures an underlying trait, such as extraversion, intelligence, or cultural values. The questions used to measure the underlying traits are typically repetitive, to ensure coverage of the trait; this fact can make the questionnaire to measure the latent beliefs unnecessarily long, which discourages respondents from answering all questions thoroughly. “Short form” questionnaires choose a subset of questions to ask respondents; this approach uses the same reduced set of questions for each individual, but different questions may be more informative for certain individuals than others. Dynamically choosing which question to ask next, based on previous information gathered about the respondent, can reduce the length and burden of the survey in a way that is personalized to the individual, while still gathering information about the relevant latent factors and estimating appropriate scores. In this paper, we present a dynamic question ordering scheme for assigning scores to respondents based on their responses to a questionnaire. Our approach is personalized to the individual answering the questionnaire and trades off the utility of having an answer to a potential new question with the cost of that question to select a cost-effective item.

We validate our approach for the application of assigning scores of cultural values to individuals. Our first validation is on a dataset of responses to an international survey (World Values Survey Association, 2015), in which we simulate question ordering on the complete dataset. Our second validation is a live deployment of our question ordering method on an online experimental platform (Reinecke & Gajos, 2015). For both validation cases, we compare our dynamic question ordering process to two baselines: a fixed-order form that asks all questions in the same order for all individuals and a short form that calculates scores with only a subset of the full question set. These validations demonstrate that a dynamic question ordering procedure can obtain estimates of equal or better quality at up to 40% burden reduction to respondents than a fixed-order long form. Additionally, the dynamic procedure allows for the possibility of asking willing respondents more questions than a fixed short form, thereby improving relative performance.

#### 4.5.1 Survey, scales, and data

In this section, we first lay out the requirements for a survey that uses dynamic question ordering to obtain low-cost scores. The remainder of this section focuses on an example of a survey that meets these requirements, which we later use to validate our dynamic question ordering method. We introduce the World Values
4.5. QUESTIONNAIRE SCORING

Survey and present the cultural values that this survey can measure. Finally, we develop an N-item short form questionnaire that can measure the same values to demonstrate that allowing each short form to be adapted to the individual (i.e., each short form can select a different subset of N items for scale estimation) can achieve more accurate estimates of the cultural values than the fixed short form questionnaire that uses the same N items from all individuals.

Requirements

Our method for dynamic question ordering for questionnaire scoring (presented in the following section) relies on historical data to learn distributions of answers. The question selection criterion we use for this problem involves the contribution that a question makes to the score calculation, so the method also requires that the survey have a score that can be calculated from responses to survey questions. Finally, while not a requirement specific to the dynamic question ordering, respondent motivation is useful for any survey, to encourage participation.

We chose the World Values Survey (WVS) for illustration of our method due to (1) the availability of a large historical set of previously completed responses, (2) the existence of previously defined and validated scales that used WVS questions to measure individuals’ cultural values, and (3) the appeal of cultural values measurement to motivate respondents to participate in our study (to receive feedback such as “Your values are most similar to those of people living in Spain and least similar to those of people living in Norway.”).

About the World Values Survey

The World Values Survey (WVS), started in 1981 as the European Values Survey, collects people’s attitudes and opinions about cultural values, across the globe [World Values Survey Association (2015)]. The questionnaire is developed by an international team of social scientists and elicits respondents’ opinions on politics (e.g., “Would you say this is a very/fairly good or fairly/very bad way of governing this country: Having a strong leader who does not have to bother with parliament and elections?”), gender roles (e.g., “Do you strongly agree or disagree: A university education is more important for a boy than a girl?”), religion (e.g., “Do you strongly agree or disagree: The only acceptable religion is my religion”), technology (e.g., “All things considered, would you say the world is better off, or worse off, because of science and technology?”), etc.

Wave 6 of the WVS was conducted from 2010 to 2014 and gathered 90,350 responses from 60 countries. Figure 4.4 shows how many responses there were from each country.

The full questionnaire has around 400 questions, but some of these questions are country-specific and not applicable to or asked of all respondents.

Measuring cultural values on the World Values Survey

Sociologists, political scientists, economists, and other social scientists have studied the relationship between economic development, democratic institutions, and cultural values (e.g., (Lipset, 1959; Dahl, 1997)). One example comes from Inglehart and Welzel (Inglehart & Welzel, 2010; Welzel, 2013), who identified two types of cultural values that distinguish societies from one another. They then developed two scales to measure these cultural values using questions from the WVS. Sacred-vs.-Secular Values indicate to what extent a respondent departs from “sacred authority” (religion, country, group norms). The secular values scale comprises four sub-indices: defiance, agnosticism, relativism, and skepticism; each sub-index has three component questions. For example, the “agnosticism” sub-index asks respondents how important religion is in their lives, how frequently they attend religious services, and whether or not they consider themselves a religious person. Obedient-vs.-Emancipative Values measure how much a person supports freedom of choice. Like the secular values scale, the emancipative values scale also has four three-item sub-indices: autonomy, equality, choice, and voice. Both measures range from zero to one, where higher values mean the person has stronger secular or emancipative values. Nations with high economic output and secular societies (e.g., Sweden) tend to score higher on both scales than nations with lower economic output and religious/traditional societies (e.g., Jordan). There is intra-society variation in individual scores that are related to income, education, and gender, but these contributing factors are not as predictive as nationality in characterizing an individual’s cultural values.
Figure 4.4: Most countries had around 1000–1500 responses to Wave 6 of the WVS. Three countries had 2500 or more responses: India (5659 responses), South Africa (3531), and Russia (2500). Three countries had fewer than 1000 responses: Trinidad (999), Poland (966), and New Zealand (841).

Inglehart and Welzel (Inglehart & Welzel, 2010; Welzel, 2013) used the WVS to identify questions and scaling formulas to measure secular values and emancipative values. Each score is calculated from responses to 12 relevant items by converting each response to a value between 0 and 1 (e.g., “Strongly agree” → 0, “Agree” → 0.33, “Disagree” → 0.67, “Strongly disagree” → 1; “Yes” → 0, “No” → 1) and then averaging these converted responses. The exact process for converting question responses to scale values is detailed in the appendix of Welzel’s book (Welzel, 2013).

Developing a short form questionnaire

To identify a relevant and representative subset of questions for the short form condition, we did a factor analysis on the 12 questions used in each scale of cultural values. Factor analysis aims to discover relationships among variables by expressing them as linear combinations of a latent space; i.e., the observed values (questionnaire responses, in our case) can be explained by the lower-dimensional space (the latent factors) (Bartholomew, Steele, Galbraith, & Moustaki, 2008). For this reason, factor analysis is often used to discover latent factors that underlie a given set of observations.

For the cultural values factor analysis, the models with four latent factors best reconstructed the item correlations from the full set of questions on a held-out validation set, so we used the four-factor models.
rather than a model with more or fewer items. Figure 4.5 is a heatmap of the factor weights for the factor analyses of the two scales (one for secular values, one for emancipative values), where the color of each cell indicates the weight of a question (y axis) on a latent factor (x axis)—warmer colors (red, orange, yellow) indicate higher factor weights, meaning that a question heavily contributes to a factor. As Figure 4.5 shows, the four-factor models make sense intuitively because questions from each sub-index (adjacent groups of questions on the y axis) score highly on separate factors. Recall that, for the cultural values scales, each sub-index consists of three related questions (such as the three questions about religion in a respondent’s life for the agnosticism sub-index of the secular values scale). For the short form questionnaire, we retained the highest-weighted question in each factor (which resulted in one question per sub-index) and discarded the rest; thus, the short form has four questions for each scale rather than 12.

Figure 4.5: Factor weights for all questions in each scale. It makes intuitive sense that related questions would rank highly on each factor.

We then compared how well this four-item short form could recreate the scores computed from the full question set (12 questions for each scale). We also compared the performance of this fixed short form to that of an optimal adaptive short form for each individual, to illustrate the potential benefit of adapting questions to respondents. The fixed short form is fixed because it uses the same four items to compute a score, for all individuals. In contrast, the optimal adaptive short form is adaptive because it allows a different subset of four items for each individual. It is optimal because it achieves the lowest error for each individual; i.e., it gives a lower bound on the error of a four-item scale. To find the optimal set of four items for each respondent, we calculated scores for all possible subsets of four items and chose the subset that yielded the score nearest the score obtained from the full set of items. This personalized process permits different subsets of items to be chosen for each respondent, but it requires knowing the actual answers for all questions.

Table 4.5 summarizes the performance of the fixed short form and the optimal adaptive short form. The first two rows give the mean absolute errors (with standard deviations in parentheses) for the short form score estimates on the Emancipative and Secular values scales. For both scales, the optimal adaptive short form yields lower-error scores, with less variance in the error among test set, than the fixed short form. The final row of Table 4.5 presents how frequently the fixed short form (from the factor analysis) resulted in the same score estimate as the optimal adaptive short form—the fixed short form is optimal on only 9.5% and 18.9% of the test set. This frequency is higher than what would be expected by chance performance of the fixed short form (since there are \( \binom{12}{4} = 495 \) potential four-item subsets from which the optimal adaptive form can choose). However, these results illustrate the potential for improvement from allowing individual respondents to use a personalized short form rather than imposing a generically accurate short form on all respondents.
Mathematically, the question selection criterion is \( \min \sum_{d} \lambda_{c_{d}} \). This approach takes previously provided information (i.e., already answered questions) into account, the utility of a candidate question \( q \) involves not only the answer for that question but also the answers to all previously answered questions. In the following notation, we will denote the answers from a \( d \)-item questionnaire in the vector \( x \in \mathbb{R}^{d} \); we will let \( x_{q} \) index the \( q \)-th item of \( x \) (i.e., the answer to question \( q \)), and \( x_{K} \) indexes all the known items of \( x \) (i.e., the answers to all previously asked questions).

Given a set of already known question responses in \( x_{K} \), computing the actual utility of question \( q \) requires knowing its answer before it has even been asked, which is not possible. Instead, we compute the expected value of the utility by taking an average of all possible outcomes (the actual utility for each possible answer \( r \in R_{q} \) to question \( q \)), weighted by the probability of those outcomes occurring. We can calculate the utility for each potential answer \( r \in R_{q} \) to question \( q \) by assuming that the answer to \( q \) is \( r \) and then calculating the utility using all the previously provided answers in \( x_{K} \), along with answer \( r \) to question \( q \): \[
E_{q}[U(x)] = \sum_{r \in R_{q}} p(x_{q} = r)U(x_{K} \cup \{r\}), \tag{4.5}
\]
where \( p(x_{q} = r) \) is the probability that the answer to question \( q \) is \( r \) and is calculated empirically from a training set (i.e., \( p(x_{q} = r) \) is the fraction that question \( q \) had answer \( r \) in the training set), and the notation \( U(x_{K} \cup \{r\}) \) means calculating the utility of an answer set consisting of all the known answers (in \( x_{K} \) and the possible answer \( r \) to question \( q \)).

The final question selection rule then balances maximizing the utility of asking a new question with minimizing the cost \( c_{q} \) of asking that new question, for all questions that have not yet been answered. Mathematically, the question selection criterion is \[
q^{*} = \arg \min_{q \notin K} (-E_{q}[U(x)] + \lambda c_{q}), \tag{4.6}
\]
where $q$ is a question, $K$ indexes known questions (i.e., questions that have already been answered), $\mathbb{E}_q[U(x)]$ is the expected utility of $q$, $c_q$ is the cost of $q$, and $\lambda \in \mathbb{R}$ is a tradeoff parameter. This tradeoff parameter $\lambda$ controls how much importance is given to question cost in the question selection: when $\lambda = 0$, cost is not considered at all and Equation 3.1 reduces to maximizing question utility alone; as $\lambda$ increases, more weight is given to question cost until the least expensive question will be chosen, regardless of its expected utility.

When we have no prior information about a respondent (i.e., no questions have been answered yet), we randomly choose the initial question.

### 4.5.3 Determining question costs

For the dynamic question ordering algorithm to be implemented, we need to know the cost of each question. Due to the importance of completion time in respondents’ decisions to answer a survey (Crawford, Couper, & Lamias, 2001), we use response time as the measure of question cost. Because the WVS dataset does not include timing information, we ran the survey on Amazon Mechanical Turk to gather response times for each question of the survey. We recruited 100 Mechanical Turk workers to take a survey which included all 24 cultural values questions. Each question was asked on a separate page, and we tracked how long each participant spent on each page before submitting an answer to that question. Our estimated completion time for the entire survey was five minutes, and participants were compensated $1.00.

Ninety-nine respondents answered all questions; the remaining person answered 23 of the 24 questions. To calculate average response times for each question, we first removed outliers, defined to be points that were more than three standard deviations away from the mean response time for that question. We assume that these unusually time-consuming instances were due to participants’ getting distracted, rather than their taking up to 25 minutes to answer a single question. Excluding these outliers, the average total time to answer all questions was 3.5 minutes (standard deviation (SD) 1.2 minutes). Average response times for each question ranged from 5.4 seconds (SD 4.0) to 37.9 seconds (SD 35.0). Figure 4.6a plots response times for the values questions in the survey. These response times are correlated with the length of the question, as Figure 4.6b illustrates: longer questions typically take longer to read and answer.

![Figure 4.6a](image1.png)  
![Figure 4.6b](image2.png)

(a) Question response times.  
(b) Question response times and question lengths.

Figure 4.6: Average response times in seconds for each question. Question response time is correlated with question length.

We used the average question response times as question costs for our experiments on question ordering.

### 4.5.4 Experiments

To illustrate the effectiveness of this dynamic question ordering method for questionnaire scoring, we implemented it for a collection of survey responses, both historical and live, and compared the performance of
our method to a standard fixed-order long form and a standard short form. We first used an established, pre-collected international survey to simulate results of partial scoring under our question ordering method. Then we deployed our questionnaire on a platform for online studies that attracts participants from across the world.

To test the benefit of our dynamic question ordering approach, we compare four conditions for collecting responses and calculating scores: (1) a fixed-order long form that asks all questions in a standardized order and calculates the scores from these responses, (2) a short form that asks a subset of questions and calculates scores from these responses, (3) a dynamically-ordered form that asks all questions, but in any order, and (4) an adaptively-ordered oracle that asks all questions and selects the next question according to which will most reduce the error of the subsequent score. For the conditions that ask all questions, we can recover the score results at any point in the process, by calculating scores using all questions asked up until that point. The fixed-order long form was determined by taking the order in which questions were asked in the standard questionnaire form and applying this order to all respondents. The short form gives all respondents the same four-item subsets of questions determined by the factor analysis to capture the most information in the individual responses. The adaptively-ordered oracle can choose questions in a different order for each respondent; it is an oracle in the sense that, since it selects the next question according to what will most reduce the actual error, it must know all of the answers to the questions before selecting which one to acquire next. The oracle provides a lower bound on score error—it is impossible for any iterative question selection scheme to achieve lower error than this oracle.

Performance metrics

The goal of our dynamic question ordering approach is to quickly obtain representative answers that result in accurate scores at low cost. To measure these quantities, we calculated three values at each point in the question asking process: the variance of responses provided so far, the error of the current score estimate compared to the final score, and the cost of all questions asked so far. We get a trajectory of the values for each of these metrics at each point in the question asking process (from when no questions have been asked to when all questions have been asked), for each test sample. We then can take the mean and variance of these metrics across all test samples to see how well the method generalizes across individuals.

Answer representativeness: Variance of responses  The variance of answers collected for an individual indicates how much the current set of answers captures the variation in an individual’s values. Because we know the answer to a question once it has been provided, we can therefore determine the actual variance of the set of answers given up to that point; i.e., this metric is the actual measure of utility for the chosen question $q^\star$. The question selection criterion in Equation 3.1 used the expected value of this quantity to choose a question to ask next, before its answer was known.

Score accuracy: Error of current score estimate  We define error as the absolute value of difference between the current estimate and the final estimate for the score. Because the final estimate is the average of the complete set of responses that have all been converted to values in the interval $[0, 1]$, we can estimate scores from partial responses as the average of responses that have been obtained up to that point. Thus, the error of an estimate $y_t$ obtained after $t$ questions have been answered is

$$\text{err}_t = |y_t - y| = \left| \frac{1}{T} \sum_{q \in K_t} x_q - \frac{1}{d} \sum_{q=1}^{d} x_q \right|,$$

where $y$ is the true score, $K_t$ indexes all questions that have been answered at point $t$ and there are $d$ total questions. The error for a single estimate is not guaranteed to decrease as more answers are provided (consider, e.g., a three-question case with answer values 0, 0.5, and 1: the final estimate is 0.5, but both question ordering schemes starting with the 0.5 answer result in the sequence of errors 0, 0.5, 0). However, due to the often related nature of the answers (since the scale is designed to measure an individual’s values along a single dimension), the errors tend to decrease as more questions are answered.
Cost of questions asked  The successive cost is defined as the sum of the costs of all questions asked up to the current time. Because all costs (item response times) are positive, this metric increases as more questions are answered.

Questionnaire scoring and question ordering on the World Values Survey dataset

We began by simulating the question asking and answering process on the World Values Survey dataset by hiding the answers to questions until they were selected in the question ordering process and “asked” and “answered.” We used 90% of the 90,350 responses as training data, to calculate empirical probabilities of question responses and to determine the fixed-order baseline. On the remaining 10% of the dataset, we considered responses individually and applied our question ordering procedure to choose questions to ask.

Figure 4.7: Results from simulation with the WVS dataset: Plots of successive error, cost, and variance as questions are asked, for our dynamic question ordering procedure as well as two baselines, a fixed-order long form and a short form that asks only four questions.

(a) Emancipative values: Successive error. (b) Emancipative values: Successive cost. (c) Emancipative values: Successive variance.

(d) Secular values: Successive error. (e) Secular values: Successive cost. (f) Secular values: Successive variance.

Figure 4.7 plots the successive error, cost, and variance as questions are asked, for our dynamic question ordering procedure as well as two baselines, a fixed-order long form and a short form that asks only four questions.

For the methods that use all questions (DQO, Oracle, and Fixed order), the lines meet at the same point once all 12 questions have been asked and answered because then all the methods have the same information. The first column of Figure 4.7 shows that the dynamically ordered questionnaire tends to have lower error than the fixed-order long form, particularly when few questions have been answered. The dynamically ordered form reaches similar error as the four-question short form once four to six questions have been answered; beyond that number, the long forms usually have lower error than the short form since they are gathering more information. The oracle almost always has the lowest error, due to its omniscient error-minimizing selection rule.

The middle column of Figure 4.7 shows that, as expected, costs are lower when higher penalties \( \lambda \) are placed on question cost in the DQO item selection rule. Particularly when cost is penalized, the dynamically
ordered questionnaire has lower cost than the short form, even when the dynamic form asked more than four questions (the number in the short form). In the emancipative values calculation, the oracle has a similar cost trajectory to the methods that do not strongly penalize question cost. In contrast, the oracle for the secular values calculation has higher cost than the other question orderings; this illustrates that error-reducing questions are sometimes more costly (as happens for the secular values), but not always so (emancipative values).

The last column of Figure 4.7 illustrates the variance of items as more questions are asked: due to the selection criterion of the dynamic procedure, the items most expected to increase variance are chosen near the beginning, with overall item variance decreasing as more questions are answered. Variance is highest when there is no or little penalty on question cost, because then the variance-maximizing term dominates the question selection rule in Equation 3.1. It makes sense that the oracle would have low variance, since the oracle sequentially selects questions that result in low-error scale estimates—the oracle first chooses the single question whose numeric value is nearest the overall score (the mean of all answers) and continues to add questions that result in an average near the overall score.

These results on the simulated dataset illustrate the usefulness—in terms of prediction quality, error, and cost to the respondent—but it remains to be seen how respondents will respond to a survey with dynamically ordered questions.

4.5.5 Questionnaire scoring and question ordering on a live deployment

To assess the effect of question ordering on survey respondents, we deployed a live implementation of a survey with our dynamic question ordering algorithm to a population of internet users. Since the simulation results showed that $\lambda = 0.01$ achieved lower-cost estimates with no decrease in performance, we used this value for the cost tradeoff parameter in the deployment. Our deployment platform was LabintheWild (Reinecke & Gajos, 2015), which recruits internet users from across the world to complete studies. LabintheWild participants are not financially compensated but instead receive personalized feedback based on their responses.

Implementing a dynamically ordered questionnaire

We implemented our dynamic question ordering algorithm on the SurveyGizmo platform. SurveyGizmo’s “professional” subscription option offers a “custom scripting” feature, using php-like SurveyGizmo functions. These functions support retrieving answers to questions, determining which questions have been answered, and jumping to a specified page of a survey, among other functionality. Our implementation in SurveyGizmo involved creating a script that would run between pages. This script (1) pulls the responses to previously answered questions, (2) calculates the expected value of each question that might be asked next (from Equation 4.5), (3) combines the expected value of each feature with its cost and selects the optimal question to ask next (Equation 3.1), and (4) jumps to the page of the next question. Each question was displayed on its own page for two reasons: (1) to allow the jump-to-page SurveyGizmo function to achieve the question ordering and (2) to allow us to calculate response times for individual questions. The full survey contains 24 questions used for determining cultural values (12 questions for each of the secular and emancipative values scales) and an additional 15 demographic questions.

On the final page of the questionnaire, we gave the respondent feedback on their cultural values scores, in terms of what country they were most and least similar to, overall and for the secular and emancipative values separately. We explained the meaning of the two scales and told respondents if they scored low, in the middle, or high on each scale and gave an example answer they gave that contributed to their score on that scale. Respondents also had an opportunity to give us feedback about their survey-taking experience or thoughts on their results in a comment box on the last page.

Results of deploying the survey to LabintheWild

Our study attracted 185 participants from 35 unique countries; 54% of respondents were from the United States. Table 4.6 summarizes participants’ countries. 67% of these participants were female, and the average age was 36.1 (SD 13.2).

\[\text{http://www.surveygizmo.com}\]
### 4.5. QUESTIONNAIRE SCORING

<table>
<thead>
<tr>
<th>Country</th>
<th>Number of responses</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>100</td>
<td>54.1%</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>9</td>
<td>4.9%</td>
</tr>
<tr>
<td>Australia</td>
<td>7</td>
<td>3.8%</td>
</tr>
<tr>
<td>Germany</td>
<td>5</td>
<td>2.7%</td>
</tr>
<tr>
<td>Japan</td>
<td>5</td>
<td>2.7%</td>
</tr>
<tr>
<td>Singapore</td>
<td>5</td>
<td>2.7%</td>
</tr>
<tr>
<td>Canada</td>
<td>4</td>
<td>2.2%</td>
</tr>
<tr>
<td>Finland</td>
<td>4</td>
<td>2.2%</td>
</tr>
<tr>
<td>South Korea</td>
<td>4</td>
<td>2.2%</td>
</tr>
<tr>
<td>Netherlands</td>
<td>4</td>
<td>2.2%</td>
</tr>
<tr>
<td>Others</td>
<td>38</td>
<td>20.5%</td>
</tr>
</tbody>
</table>

Table 4.6: Participant countries for the LabintheWild deployment.

Figure 4.8 plots the error, cost, and variance metrics for the LabintheWild results, for the question orders given by DQO (with $\lambda = 0.01$ for the deployment) and the two baselines: fixed-order and short form. As in the previous section’s WVS simulation, the dynamically ordered procedure results in score estimates that cost less than the fixed-order forms while maintaining similar accuracy. While these overall trends hold across both WVS and LabintheWild validations, there are some differences. In the WVS dataset, the emancipative values error trajectory (Figure 4.7a) was higher for the fixed-order form (dashed black line) than for the DQO form (solid blue lines). In the LabintheWild dataset, however, the fixed-order form and DQO form yield similar emancipative values error trajectories. This behavior may be due to the difference in score distribution among the responses in the WVS dataset (emancipative values: $0.41 \pm 0.18$, secular values: $0.36 \pm 0.17$) and the responses in our LabintheWild dataset (emancipative values: $0.76 \pm 0.15$, secular values: $0.60 \pm 0.16$). These differences are caused by the overrepresentation of the United States in our deployment.

On the final page of the survey, on which respondents received their results and explanations, there was an optional free-response question where they could give us feedback about their survey-taking experience. Most respondents left this field blank. The few who did respond typically commented that they found the survey “interesting” or “very interesting.” The respondents gave no indication that they found the dynamically ordered form confusing or disorienting. While the questionnaire did have related and similar questions that would have been presenting sequentially in a fixed-order form (e.g., asking religion-related questions in a row), it is possible that these questions had low enough cognitive burden (Tourangeau, 1984) that it did not trouble participants to skip among unrelated questions. This aspect of the feasibility of a dynamic question ordering method is likely application-dependent—some questionnaires have sections of questions that are very related and easier for respondents to answer if such questions are grouped together.

#### 4.5.6 Conclusion and Future Work

Being able to quickly understand users’ questionnaire responses, with low burden on users, is important for researchers who want to understand participants’ attitudes and values. In this paper, we demonstrated that, unlike previous approaches that try to decrease cost uniformly across all participants (by asking everyone the same subset of questions in a short form), allowing the subset of questions to be adapted to the individual can improve performance. We developed a method that leverages previously provided information from a user to select which question is best to ask next, trading off the expected usefulness of that question with its cost. Our experiments, both simulations using a previously collected dataset of over 90,000 responses as well as a live deployment where participants filled out a dynamically-ordered form, illustrated the cost-saving benefits of this adaptive approach: in both settings, our method resulted in lower-cost score estimates that were similar to or better than the fixed order baseline in terms of error. Furthermore, the dynamic procedure allows the questionnaire administrator to continue asking questions beyond a short form’s capacity, when a respondent is willing to spend more time or effort answering questions.

In this application, we validated our dynamic question ordering approach on a survey with two scales to calculate two types of cultural values of the respondent: secular and emancipative values. We treated
CHAPTER 4. PREDICTION-GUIDED QUESTION ORDERING

Figure 4.8: Results from live deployment on LabintheWild: Plots of successive error, cost, and variance as questions are asked, for our dynamic question ordering procedure as well as two baselines, a fixed-order long form and a short form that asks only four questions.

these two scales individually, first dynamically ordering the questions associated with the first scale and then dynamically ordering the questions associated with the second scale. However, a more nuanced approach for such multiple-scale questionnaires could choose items from the different scales in tandem, rather than doing one scale at a time (as we did). Often, scales in the same survey are related, and we can take advantage of these relationships to achieve even better cost performance. We could incorporate multiple scales into the item selection rule by learning relationships between scores and all questions (not just the questions that explicitly define each score) and then modifying the utility function so that a single question’s utility includes its effect on all scales. Incorporating this additional information into the process will further reduce the burden on respondents (since all questions can influence all score estimates), particularly for longer dynamically ordered questionnaires compared to the longer fixed orders.

Another extension is to use some prior information to initialize the first question being asked, rather than randomly selecting it (as we did, when no questions had been answered). For example, an online survey could identify a respondent’s location (country or region) from their IP address and initially select a question that is likely most to differentiate respondents in that country. Furthermore, if the questionnaire is part of a series (since many online research sites offer multiple questionnaires, e.g., Reinecke & Gajos 2015), prior information from a respondent’s previous surveys can inform the choice of first question.

Additionally, we can consider information that can be automatically collected while the user is taking the survey, along with the information the user provides. For example, paradata collected as the respondent answers an online survey (e.g., time spent on page, mouse clicks Kaczmirek 2008) can be used to model user engagement, with subsequent survey action taken to increase user engagement and response rates M. P. Couper et al. 2010.
Chapter 5

Data collection-focused question ordering

In the previous chapter, the goal was to gather information for making predictions and we saw that not all information was necessary to make a good-quality prediction. Therefore, dynamically ordering features to acquire could move the most useful (in terms of high utility and low cost) to the beginning of data collection and not expend the resources to acquire all features. In contrast, surveys are typically concerned with gathering complete information from a population. However, survey respondents are not always willing to take the time or effort to fill out complete surveys, as evidenced by declining response rates (e.g., (Porter, 2004; Shih & Fan, 2008)) and breakoffs partway through surveys (e.g., (Horwitz, Tancreto, Zelenak, & Davis, 2012)). Thus, there are two scenarios for survey-focused dynamic question ordering that we consider: (1) modeling user engagement and ordering questions in ways that will keep users motivated to complete the survey, and (2) collecting the information that most characterizes the respondent so that if they do drop out of the survey, imputed values for their unanswered questions will be accurate. Furthermore, some surveys do have prediction goals (e.g., predicting if a respondent is in the labor force, for the employment rate estimate in the Current Population Survey (U.S. Bureau of the Census, 2006)), so the prediction-guided question ordering from the previous chapter could also apply to these scenarios.

Any statistics-based approach to dynamic question ordering of the sort we consider here would seem to run counter to traditional arguments that questionnaires should have a fixed structure for all respondents and when the same quantities, e.g., unemployment or poverty, are measured by surveys over time. Just over thirty years ago, the cognitive aspects of survey methodology (CASM) movement, (e.g., (Jabine, Straf, Tanur, & Tourangeau, 1984; Sudman et al., 1996; Tanur, 1992)), made the argument that this traditional approach to survey design shackled respondents and often prevented them from providing the very answers that the survey methodologists sought for their questions, (e.g., (Suchman & Jordan, 1990; Tanur, 1992)). We believe our approach reopens the door to the arguments raised by that movement, but in a very different manner, and somehow survey statisticians will ultimately need to blend the lessons from the CASM movement with the needs for cost-driven dynamic ordering.

5.1 Moving from population-level adaptation to respondent-level adaptation

Typical adaptive survey designs are adaptive at the population level—they choose what survey protocol to use between phases of data collection (e.g., (Groves & Heeringa, 2006)); they decide how much effort to expend on making contact for a survey (e.g., (Beaumont, Boci, & Haziza, 2014; Calinescu, Bhulai, & Schouten, 2013)), sometimes using paradata about contact attempts (e.g., (M. Couper & Wagner, 2012; Lundquist & Sarndal, 2013; Sauermann & Roach, 2013)); they assign subsamples to fixed subsets of questions (e.g., (Gonzalez & Eltinge, 2008)). Dynamically ordering questions for a single respondent, based on their previously-answered questions, can also be considered an adaptive survey design. It is this aspect of respondent-level adaptation that we are studying.
As in the previous chapter on predicted-guided question ordering, we consider iteratively asking questions to a single respondent, trading off how useful the question is against how costly its answer is to obtain. Whereas in Chapter 4 a feature’s utility reflected how useful it was to the subsequent prediction, here we define a question’s utility in terms of how it influences the respondent’s engagement or the quality of imputations for unknown values.

5.1.1 Using paradata to model user engagement

Information collected about the survey process, known as paradata (originally “process data” (M. P. Couper 1998)), can indicate a respondent’s engagement with and understanding of the questionnaire they are answering. Paradata can include interviewer-collected data like response times, changing answers, returning to earlier questions in the survey, clicking help buttons, etc. Past work in paradata analysis has focused on how paradata reveal respondents’ survey-taking process, how paradata can identify usability issues in surveys, and how paradata can inform population-level adaptive design. For example, Bassili and Fletcher (1991) and Heerwegh (2003) showed that participants who take longer to answer questions about attitudes also tend to change their minds when presented with counterarguments to their original response; they conclude that people with less attitude stability need more time to come up with an answer than people who are already confident in their attitudes. Paradata about users’ interactions can also reveal that a survey instrument is not designed well, as Healey (2007) illustrated by finding that drop-down menus in online surveys increased item response times; question length (e.g., (Bassili & Scott, 1996; Yan & Tourangeau, 2008)) and complexity (e.g., (Wagner-Menghin, 2002; Yan & Tourangeau, 2008)) also increase response times. Paradata have been used for adaptive survey design too; in particular, for predicting the likelihood that a unit will be interviewed on a next contact attempt (Groves & Heerling, 2006; M. Couper & Wagner, 2012) and for analyzing differences in population estimates as contacts are made (Lundquist & Särndal, 2013).

5.2 Estimating population-level parameters from dynamically-ordered questionnaires

The end goal of most surveys is to obtain population-level estimates from responses. When data are missing from surveys, researchers often use weighting adjustments to account for unit nonresponse (e.g., (Brick & Kalton, 1996; Brick, 2013)) and imputation to account for item nonresponse (e.g., (Rubin, 2004; Brick & Kalton, 1996)). In our experiments, we will use the missing data mechanisms used for each survey application to compare the population-level estimates obtained from a dynamically-ordered survey with those obtained from the traditional fixed-order survey.

5.3 Applications and experiments

We consider two large-scale surveys conducted by the U.S. Census Bureau: the American Community Survey (ACS) and the Survey of Income and Program Participation (SIPP). Because such surveys have multiple goals and complex designs, we begin by looking at each aspect of dynamic question ordering separately: ordering questions to predict a survey outcome, ordering questions to improve user engagement and response rates, and ordering questions to improve imputation quality of unanswered questions.

5.3.1 Survey collection with the ACS

For the mandatory American Community Survey (ACS), the goal is to gather complete statistics on the U.S. population, and follow-up with nonrespondents is expensive. Each year 3.54 million households receive mailed surveys to answer anywhere between 77 and 347 questions, depending on the number of household occupants (U.S. Bureau of the Census, 2016). The survey takes, on average, 40 minutes to complete and 54% of homes return theirs (U.S. Bureau of the Census, 2014a). The Census Bureau calls nonrespondents for telephone interviews and then samples nonrespondents for home interviews. Each in-person case takes 134 minutes; in 2012 this amounted to 129,000 person-hours per month (Griffin & Nelson, 2014). In addition to being expensive, in-person interviews can also bias survey results due to higher weights assigned to those
subsampled respondents (U.S. Bureau of the Census, 2014a). The Census Bureau tested shifting the mail survey online and found similar data quality for internet and mail return (Horwitz et al., 2012). Furthermore, while overall response rates were similar, online surveys had higher item response rates for earlier questions and more blank responses for later questions than paper surveys (Horwitz et al., 2012). Dynamically ordering survey questions in the online form could ensure that even if households do not complete the survey, they answer the most informative questions before breaking off.

The online mode for the ACS also collects paradata as respondents complete the survey. These paradata include clicked links (including navigation buttons, responses, help buttons), timestamps, field values, errors, invalid logins, timeouts, logouts (Horwitz et al., 2012). Such paradata could be used to model user engagement, understanding, and willingness to respond, as another component for dynamic question ordering to increase response rate.

The unique design of the ACS will allow us to explore three facets of our DQO framework. First, we will determine the individual question orderings to optimize response quality, measured by survey completion and imputation quality. Second, we will use paradata collected in the online mode of the ACS to model user engagement, measured by response rate. Third, we will consider the effect of this respondent-level adaptation on population-level estimates.

5.3.2 Survey collection with the SIPP

Conducted by the U.S. Census Bureau, the Survey of Income and Program Participation (SIPP) collects data on income, employment, and social program participation and eligibility from households (U.S. Bureau of the Census, 2014b). The SIPP is designed as a longitudinal national panel survey, where each panel is a representative sample of 14,000 to 52,000 households, contacted yearly for three to five consecutive years. Each household interview is conducted in person, via a computer-assisted personal interviewing (CAPI) instrument, and aims to get self-reports from all household members at least 15 years old. In addition to demographic information, interviews ask respondents for their participation in various social programs, financial situation, and employment status, in the previous calendar year. The chief goal of SIPP is to understand household program eligibility and participation and to assess the effectiveness of social programs like Supplemental Security Income, Supplemental Nutrition Assistance Program, Temporary Assistance for Needy Families, and Medicaid. Using participation in each program of interest as the prediction of interest could guide a DQO approach.

The unique design of SIPP will allow us to explore four facets of our DQO framework. First, we will determine individual question orderings to optimize response quality (survey completion and imputation quality). Second, we will determine question orderings to optimize the quality of predictions of respondents’ participation in each social program of interest; it is very likely that not all questions are necessary to predict participation, and DQO could reduce respondent burden (interview length) while still collecting the information the Census Bureau needs. Third, we will consider the effect of this respondent-level adaptation on population-level estimates. Finally, we will use the longitudinal nature of SIPP: data from prior interviews can be used as “background” in subsequent interviews.

In our experiments, we use the SIPP Synthetic Beta (SSB), a synthetic dataset created by first multiply-imputing missing values in the SIPP Gold Standard File and then multiply-imputing replacement values for actual responses to preserve privacy (Benedetto, Stinson, & Abowd, 2013).
Chapter 6

Proposed work

The prediction-focused question ordering presented in Chapter 4 has been completed. Remaining work is for the survey-taking case, from Chapter 5 using data from the SIPP Synthetic Beta (SSB) and the American Community Survey (ACS).

6.1 Timeline

- October 2016: proposal
- November 2016: prediction-focused question ordering on SSB, using participation in food stamps as the predictor of interest
- December 2016: imputation-focused question ordering in SSB
- January 2017: using the longitudinal nature of SIPP to guide question ordering in subsequent interviews for the same household
- February 2017: start working with ACS data
- March 2017: imputation-focused question ordering in ACS
- April 2017: modeling respondent engagement using paradata in the online ACS
- May 2017: finish any outstanding work
- June 2017: writing
- July 2017: defense
References


References


Min, C., Yoo, C., Hwang, I., Kang, S., Lee, Y., Lee, S., ... Song, J. (2015). Sandra helps you learn: The more you walk, the more battery your phone drains. In *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing* (pp. 421–432).


References


Appendix A

Algorithms

Algorithm 1 Estimating values $z$ for still-unknown features

Input: $X \in \mathbb{R}^{n \times d}, x \in \mathbb{R}^d, \mathcal{K} \subseteq \{1, \ldots, d\}, k \in \mathbb{Z}^+$

Output: $z \in \mathbb{R}^d$

1: function ESTIMATE_FEATURES($X, x, \mathcal{K}, k$) 
2: $z_{\mathcal{K}} \leftarrow x_{\mathcal{K}}$ \hspace{1cm} $\triangleright$ Copy over the known features
3: $I \leftarrow \text{get_knn}(X, \mathcal{K}, z_{\mathcal{K}}, k)$ \hspace{1cm} $\triangleright$ Index $z_{\mathcal{K}}$’s $k$NNs
4: for $f \in \{1, \ldots, d\} \setminus \mathcal{K}$ do 
5: $z_f \leftarrow \text{mean}(X_{I, f})$ \hspace{1cm} $\triangleright$ Estimate $z_f$ from $k$NNs’ values for feature $f$
6: end for
7: return $z$
8: end function

Algorithm 2 Calculating the expected prediction uncertainty for each candidate feature

Input: $\mathcal{K} \subseteq \{1, \ldots, d\}, z \in \mathbb{R}^d$, feat_ranges, feat_proportions, model

Output: $E \in \mathbb{R}^d$

1: function EXPECTED_UNCERTAINTY($\mathcal{K}, z$, feat_ranges, feat_proportions, model) 
2: $E \leftarrow 0^d$
3: for $f \in \{1, \ldots, d\} \setminus \mathcal{K}$ do 
4: $R \leftarrow \text{feat_ranges}\{f\}$, $u \leftarrow 0^{|R|}$ \hspace{1cm} $\triangleright$ For each value $f$ can take on
5: for $\ell \in \{1, \ldots, |R|\}$ do 
6: $\bar{z} \leftarrow z$, $\bar{z}_f \leftarrow R_\ell$ \hspace{1cm} $\triangleright$ $f$-th feature is assigned
7: $u_\ell \leftarrow \text{CALCULATE_UNCERTAINTY}(\bar{z}, \text{model})$
8: end for
9: $p \leftarrow \text{feat_proportions}\{f\}$
10: $E_f \leftarrow p^T u$ \hspace{1cm} $\triangleright$ Expectation of prediction uncertainty
11: end for
12: return $E$
13: end function
Algorithm 3 Dynamically choosing a question ordering $\mathcal{A}$ and making a sequence of predictions $\hat{y}$ at the current feature values and estimates as feature values are provided

**Input:** $X \in \mathbb{R}^{n \times d}$, $x \in \mathbb{R}^d$, $\mathcal{K} \subseteq \{1, \ldots, d\}$, feat ranges, feat proportions, $\lambda \in \mathbb{R}$, $c \in \mathbb{R}^d$, model

**Output:** $\mathcal{A} \subseteq \{1, \ldots, d\}$, $\hat{y} \in \mathbb{R}^{d\mid \mathcal{K} \mid + 1}$

1. function DQO\_ALL($X$, $x$, $\mathcal{K}$, $k$, $\delta$, $\alpha$, feat ranges, feat proportions, $\hat{\beta}$, $\hat{\sigma}^2$, $\lambda$, $c$)
2. $\mathcal{A} \leftarrow \{\}$, $\hat{y} \leftarrow \{\}$
3. for $i \in \{1, \ldots, d - |\mathcal{K}|\}$ do
4. $z \leftarrow$ ESTIMATE\_FEATURES($X$, $x$, $\mathcal{K}$)
5. $\hat{y}_i \leftarrow$ PREDICT($z$, model) $\triangleright$ Predict on features and estimates
6. $E \leftarrow$ EXPECTED\_UNCERTAINTY($X$, $\mathcal{K}$, $z$, feat ranges, feat proportions, model)
7. $f^* \leftarrow \arg \min_{f \in \mathcal{K}} (E_f + \lambda \cdot c_f)$
8. $\mathcal{A} \leftarrow \mathcal{A} \cup \{f^*\}$, $\mathcal{K} \leftarrow \mathcal{K} \cup \{f^*\}$
9. $z_{f^*} \leftarrow x_{f^*}$ $\triangleright$ Ask and receive value for $f^*$
10. end for
11. $z \leftarrow$ ESTIMATE\_FEATURES($X$, $x$, $\mathcal{K}$)
12. $\hat{y}_{d\mid \mathcal{K} \mid + 1} \leftarrow$ PREDICT($z$, model) $\triangleright$ Make final prediction
13. return $\mathcal{A}$, $\hat{y}$
14. end function