Test Time Feature Ordering with FOCUS: Interactive Predictions with Minimal User Burden

Kirstin Early[1], Stephen E. Fienberg[1,2], Jennifer Mankoff[1,3]
[1]Machine Learning Department, [2]Department of Statistics, [3]Human-Computer Interaction Institute
Carnegie Mellon University, Pittsburgh, USA
{kearly, fienberg, jmankoff}@andrew.cmu.edu

ABSTRACT
Predictive algorithms are a critical part of the ubiquitous computing vision, enabling appropriate action on behalf of users. A common class of algorithms, which has seen uptake in ubiquitous computing, is supervised machine learning algorithms. Such algorithms are trained to make predictions based on a set of features (selected at training time). However, features needed at prediction time (such as mobile information that impacts battery life, or information collected from users *via* experience sampling) may be costly to collect. In addition, both cost and value of a feature may change dynamically based on real-world context (such as battery life or user location) and prediction context (what features are already known, and what their values are). We contribute a framework for *dynamically trading off feature cost against prediction quality at prediction time.* We demonstrate this work in the context of three prediction tasks: providing prospective tenants estimates for energy costs in potential homes, estimating momentary stress levels from both sensed and user-provided mobile data, and classifying images to facilitate opportunistic device interactions. Our results show that while our approach to cost-sensitive feature selection is up to 45% less costly than competing approaches, error rates are equivalent or better.

Author Keywords
Online data collection; interactive machine learning; cost-based dynamic question ordering

ACM Classification Keywords
I.5.2. Pattern recognition: Design methodology: Feature evaluation and selection.

INTRODUCTION
Online data collection from individuals can provide them with personalized, timely predictions 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 these costs while still making useful predictions. We develop an approach to test time Feature Ordering with Cost and Uncertainty Score (FOCUS) that trades off the expected *utility* of the next prediction, were we to have any of the yet-unanswered feature values, with the cost of acquiring that feature. FOCUS sequentially requests feature values to make useful, confident predictions with the resources users are willing and able to provide.

We validate our approach in the context of three datasets drawn from related existing work. The first is a U.S. Government-collected dataset about household energy use [46]. The second is a research dataset that collected both sensed and user-provided data from college students over the course of one ten-week academic term, including self-reports on stress levels [47]. The third is a dataset used to validate a mobile application that can identify devices (*e.g.*, printers, projectors) in photographs to support lightweight opportunistic interaction without forcing users to manually install drivers [9]. On all three, we demonstrate that prediction quality is not significantly worse than a standard fixed-order baseline, while costs are decreased.

BACKGROUND
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 [7].

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

UbiComp '16, September 12-16, 2016, Heidelberg, Germany

© 2016 ACM. ISBN 978-1-4503-4461-6/16/09 $15.00
DOI: http://dx.doi.org/10.1145/2971648.2971748

test time feature acquisition is to speed up prediction (*e.g.*, [21, 42, 49]). In other domains, there is an allowable test time budget of costs other than time, such as user burden of providing information (*e.g.,* [12]) or the resulting loss in privacy from disclosing information (*e.g.,* [33]).

Feature selection—*i.e.,* choosing at training time a relevant subset of features to include in a model [19]—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.*, [1]) 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*.

Cost & order at training time; # at test time

One approach to test time feature selection is learning *sequences* of features to add. For example, Strubell *et al*. [42] 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 [51], 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.

Cost at training time; # & 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.*, [20, 21, 39, 41, 49]). 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.,* [20, 21, 39, 49]).

This is a fairly general approach that has been used quite widely. For example, Shi *et al*. [41] 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" [26] is improved by eliciting the most relevant preferences from the user to minimize burden (*e.g.*, [18, 43]). 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.* [18]). 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.

Cost & # & 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*. [33] 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.

This paper presents a generalization of preliminary work focused on adaptive survey design, particularly for personalized predictions [12]. The dynamic question ordering (DQO) algorithm presented in that work is designed to minimize the prediction interval width in the context of regression and was applied to the energy use dataset also presented in this paper. This publication presents FOCUS (Feature Ordering with Cost and Uncertainty Score), an extension of DQO that (1) uses metrics for cost rather than just assigning a binary value (low or high), (2) accounts for costs that may change dynamically based on context, (3) supports a wider variety of supervised machine learning algorithms, and (4) validates the approach in multiple datasets rather than just one.

Summary of key attributes

Table 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.

Paper	Prediction Problem	Cost Metric	Test-Time	Utility Function	Domain
FOCUS	C,R	FS	C/O/#	U	Ubicomp/ Mobile/ Energy
[20]	C	FS	X	M	—
[21]	C	X	X	A	NLP
[42]	C	X	#	M	NLP
[41]	C	X	X	A	Structured prediction
[49]	C	FS	#	A	Structured prediction
[51]	C	FS	X	A	LTR, MNIST
[39]	C	FS	O/#	M	Knowledge-on-Demand
[33]	C	FS	O/#	U	Information disclosure
[36]	C	FS	#	A	—
[18,43]	C	X	X	A	Recommender Systems

Table 1 A summary of previous work in test time feature acquisition and our method (FOCUS, top row). The aspects of test time feature selection we compare are: whether the type of prediction problem is classification (C) or regression (R); whether the cost metric for features is Feature-Specific (*i.e.*, different features can have different costs; X means no cost metric); what is determined at test time (**C**ost, **O**rder, **#** of features); the utility function used (**A**ccuracy or **U**ncertainty or **M**argin from next closest class); in what domains the algorithm has been applied.

Table 1's Column 1 (**Prediction Problem)** shows that algorithms may predict discrete values (classification) or continuous values (regression); the majority of 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.*, [16]), 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.*, [21, 41, 49, 51]) and subsequent prediction uncertainty (*e.g.*, [33]). Finally, the **Domain** column shows where each relevant approach was applied.

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).

OVERVIEW OF FOCUS

We present an algorithm, FOCUS, that has all of these key properties. We build on the DQO algorithm [12] and extend it to include classification and richer metrics for feature costs, including context-dependent costs which are not evaluated until test time. 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*).

As shown in Figure 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.

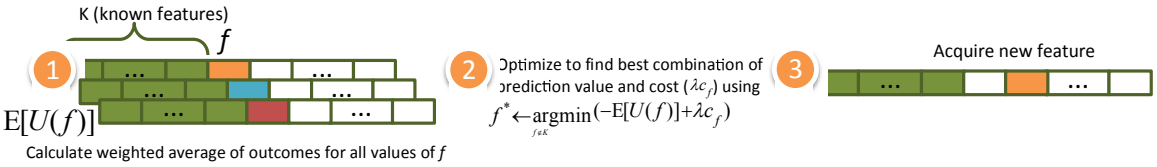

Figure 1 FOCUS iteratively increases the set of known features. **Step 1:** Given a set of known features, it first calculates the expected prediction value and cost of acquiring each unknown feature. This is repeated for all unknown features. **Step 2:** FOCUS optimizes for the best combination of prediction value (as calculated in Steps 1 and 2) and cost. **Step 3:** The best next feature is acquired. Now a prediction can be made, or FOCUS can repeat this process.

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 1, FOCUS sequentially estimates the value of each feature (Step 1) and selects a next feature to ask (Steps 2 and 3).

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 $\mathbb{E}[U(f)]$. 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 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.*, [51]) or dynamically constructed at test time (*e.g.*, [17]). 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.*, [38]). However, in the end, FOCUS is agnostic about how predictions are made and how *U* is defined.

Calculating the Expected Prediction Utility of a Feature f

Given a way to calculate $U(f)$, $\mathbb{E}[U(f)]$ 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*:

$$\mathbb{E}[\boldsymbol{U(f)}] = \sum_{r \in R} \boldsymbol{p(z_f = r) U(z_{f:=r})}, \qquad \textbf{Equation 1}$$

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*.

This process is then repeated for all unknown features.

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 1, trades off the expected utility of the next prediction, for each candidate feature, with the cost of that feature:

$$\boldsymbol{f^\star = \arg\min_{f \notin K} (-\mathbb{E}[U(f)] + \lambda c_f)}, \qquad \textbf{Equation 2}$$

where $\mathbb{E}[U(f)]$ is the expected utility calculated in Step 1 of FOCUS, and λ controls how much weight we give to the cost *c* for each feature.

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.

VALIDATION

To demonstrate the usefulness of FOCUS for cost-effective interactive predictions, we implement 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 battery drain from turning on mobile sensors and the cost of interrupting the user to ask for feature values. In this example, we also consider *context-dependent* costs (the cost of turning on a sensor at the expense of draining the battery is not 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.

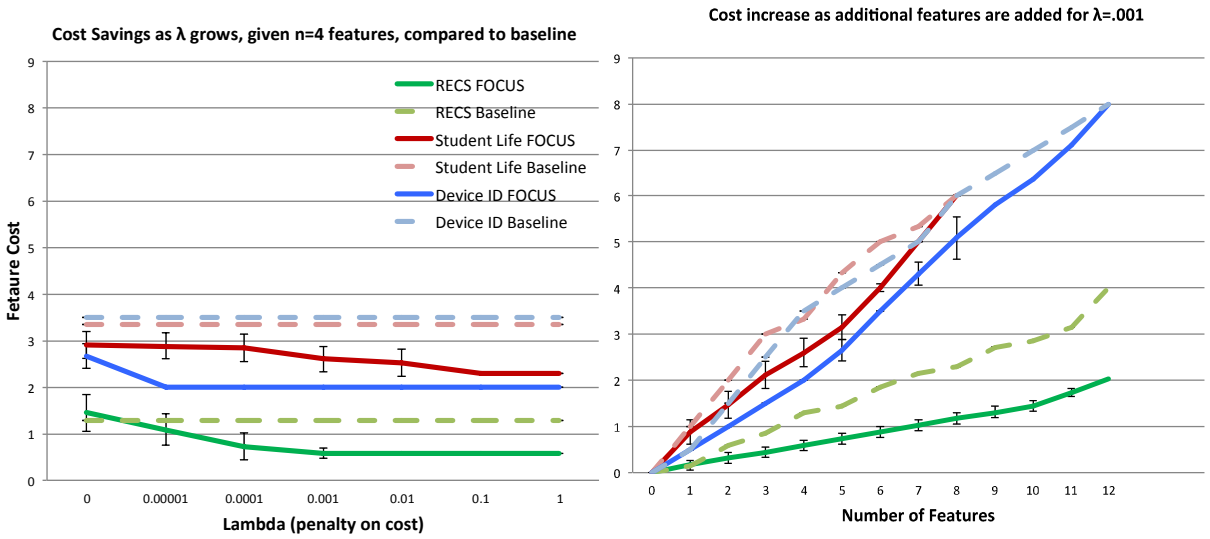

Figure 2 Charts showing impact on cost (y axis) of λ and number of features. (*Left*) As λ increases, cost savings from FOCUS increase, *without significant loss of accuracy.* Dashed lines show baseline cost performance. (*Right*) FOCUS (solid lines) ensures that cost increases more slowly than the baseline algorithm, with λ =.001. Note that the Student Life and Device ID sets converge because there are only 8 and 12 total features; RECS converges at 30 features (not shown).

For each of these applications, we compare the quality of successive predictions obtained with FOCUS, with a variety of cost penalties λ, to that of a fixed-order baseline, which we call *Fixed Selection*. This baseline acquires features in the order of forward selection [45] 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 2.

In all three applications, we use imputation to predict with partial features as follows: Using *k*NN [8], 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.

We optimize for *prediction certainty* in each of our validation datasets due to past work demonstrating the importance of providing people with certainty of predictions to help them make decisions (*e.g.*, [22]). 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 [48]. A narrower prediction interval corresponds to a more certain prediction. For classification, certainty can often be formulated as distance from the decision boundary. Other example metrics for prediction certainty are discussed in more detail in the applications in the next section. Using this metric, we can estimate certainty of the next prediction, if a feature were available. In our equation for test-time feature ordering (Equation 2), certainty is easily replaced with other metrics (*e.g.*, prediction error).

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 2 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 shows that, as the cost penalty λ increases, the cost savings of FOCUS over the baseline also increases. As expected, increasing the cost tradeoff parameter λ favors asking inexpensive features near the beginning of the test time feature acquisition process. The right plot in Figure 2 shows that, for a fixed λ 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 2 lists these values for FOCUS with seven different values of cost penalty λ, ranging from zero to one, and for the baseline. As

Method		RECS			StudentLife			Device ID		
		Cost	*Uncert.*	*Error*	*Cost*	*Uncert.*	*Error*	*Cost*	*Uncert.*	*Error*
FOCUS, λ =	0	1340.251†	**41.376***	**12.457**	**71.182***	**22.897**	5.770	**94.788***	1.935	4.815
	.00001	**1175.420***	**41.375***	**12.444**	**70.924***	**22.898**	5.769	**87.647***	1.939†	5.241
	.0001	**936.338***	**41.371***	12.649†	**70.030***	**22.898**	5.771	**87.601***	1.940†	5.259
	.001	**891.130***	**41.430***	12.578†	**66.704***	**22.899**	5.775†	**87.638***	1.936	5.296†
	.01	**885.000***	41.446	12.704†	**63.303***	**22.929**	5.795†	**82.001***	1.936	5.463†
	.1	**885.000***	41.446	12.704†	**61.303***	**22.958**	5.800†	**80.547***	1.936	5.574†
	1	**885.000***	41.446	12.704†	**61.303***	**22.958**	5.800†	**80.000***	1.926	5.333†
Baseline		1250.000	41.447	11.722	81.000	22.898	5.705	105.000	1.928	4.796

Table 2 Areas under the curve for the cost, uncertainty, and error metrics from **FOCUS**, with a variety of cost penalties λ, and the **Baseline** (*Fixed Selection*). Ideally, FOCUS will have lower cost and similar uncertainty as Baseline. This is shown in black. **Bold** numbers (marked with *) show where FOCUS was significantly lower than baseline at the $\alpha = 0.05$ level. Red numbers (marked with †) indicate where FOCUS did significantly worse than baseline (higher cost, uncertainty, or error) at the $\alpha = 0.05$ level.

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 λ 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 λ always has significantly lower cost than the baseline. Prediction uncertainty is sometimes lower with FOCUS than baseline (particularly for low values of λ). The metric that suffers the most with FOCUS, relative to the baseline, is error because it was not included in the optimization.

Personalized Energy Estimates 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 [10]. 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% [35]), 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) [46], 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 [44]. Household features may include macroeconomic indicators, as well as occupant-specific features (*e.g.*, [11, 25]). For example, Douthitt [11] examines fuel consumption for space heating in Canada, using household-specific values for occupant demographics, the housing structure, and fuel cost. Kaza [25] 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 *et al.* [12], 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

	Reason	Example Feature
0	Extractable from listing	Number of bedrooms
1	User probably already knows	If someone stays home during the day
2	Might have to check current home	Size of TV
3	Could find in listing or call for easy answer	Washing machine in home
4	Look up online	Year housing unit was built
5	Easily visible during visit	High ceilings
6	Requires effort to find out during visit	Type of glass in windows
7	Requires visit + look something up	Age of heating equipment

Table 3 Cost categories for RECS dataset.

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 3 lists the cost categories and an example feature in each.

Experimental Setup

We first divide RECS into training (90%) and testing (10%) sets. At training time, we use forward selection [45] 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.

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 λ except 0 with equivalent or better uncertainty (Table 2). For λ=.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 [22].

Momentary Stress Predictions in College Students

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 [47]. 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 [32], with individual-level health and societal-level economic consequences [24]. College students face a unique and significant type of stress, partly because of their transition between dependent child and independent adult (*e.g.*, [37]).

It is possible to estimate stress from *physiological* and *physical* signals, like skin conductance, brain activity, and pupil dilation (*e.g.*, [40]); 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.*, [15, 23, 31]).

For example, Affective Health [15] 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 [47] 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 3 illustrates the number of stress reports and average stress levels for each participant.

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

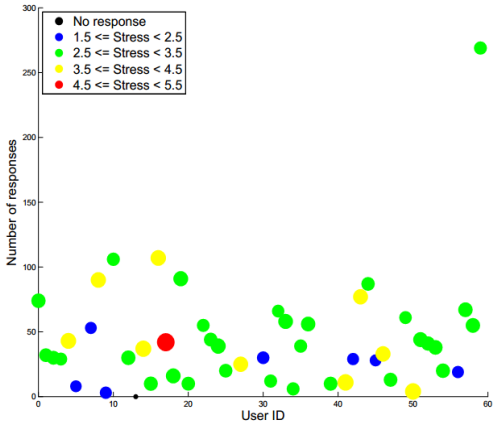

Figure 3 StudentLife participants' number of stress reports (*y* axis) and average stress (color, marker size).

Feature	Regression weight	p-value
Intercept	-0.5401	0.1471
Time of day	-0.0295	0.7950
Sleep	0.6187	0.2281
Exercise	-0.1151	0.2912
Time to deadline	-0.0401	0.8559
# of deadlines	0.2625	0.1137
Currently moving?	-0.0226	0.8868
Currently silent?	0.0479	0.6125
Currently dark?	-0.0864	0.2117
Currently charging?	0.0492	0.5649

Table 4 Regression weights from our linear model to predict stress from sensed and user-provided data in StudentLife.

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.*, [14, 29])) 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.*, [23, 31]), 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.

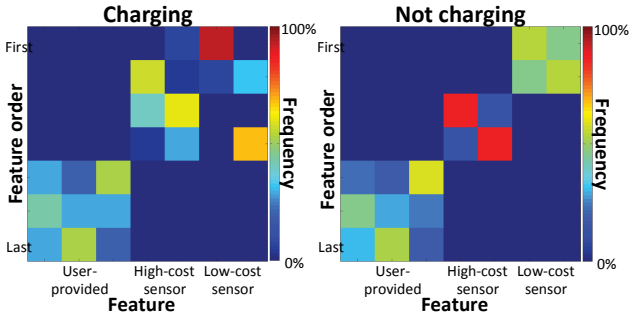

Figure 4 Context-dependent costs, with $\lambda = 1$. Color indicates frequency; x axis indicates feature type; y axis represents feature order.
(*Left*) When the phone is charging, acquiring sensor features is no longer expensive, and so sensed features (last four columns) are requested more uniformly (top right corner).
(*Right*) 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 top right corner).

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 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 2 (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 2. Prediction certainty decreased slightly for λ=.001 after n=5. On average, FOCUS yielded 23% savings in cost compared to the baseline as features were added to the prediction.

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 shows how frequently each feature was chosen in each position, for the charging and noncharging contexts, when cost penalty parameter λ equals 1. The battery-draining sensors are in the last four columns. When the phone is charging (left), 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 (right), features tend to be asked in order of increasing cost—most 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.

Image Classification for Opportunistic Interactions

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) 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 [8]. 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.*, [3]), external cameras (*e.g.*, [5] with Kinect), and magnetometers (*e.g.*, [50]). For the case of smartphone-taken images, past work explores directly identifying appliances labeled with fiducial markers (*e.g.*, [27]) or using image recognition to identify a pictured device (*e.g.*, [6]). Snap-To-It [9] 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 [28]. 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 Metric

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 SnapToIt 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 [4]. 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 2 (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 λ=.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*. [9] 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.

DISCUSSION & CONCLUSIONS

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 λ. 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 λ; 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.*, [13, 30]), or the expected time-to-next-charge (*e.g.*, [2, 34]). 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.

ACKNOWLEDGMENTS

This work was supported by NSF grants IIS-1217929 and SES-1130706, as well as the Siebel Foundation.

REFERENCES

1. Miguel Ballesteros & Bernd Bohnet. (2014). Automatic feature selection for agenda-based dependency parsing. In *COLING* (pp. 794-805).
2. Nilanjan Banerjee, Ahmad Rahmati, Mark D. Corner, Sami Rollins, & Lin Zhong. (2007). *Users and Batteries: Interactions and Adaptive Energy Management in Mobile Systems* (pp. 217-234). Springer Berlin Heidelberg.
3. Michael Beigl. (1999). Point & click-interaction in smart environments. In *Handheld and Ubiquitous Computing* (pp. 311-313). Springer Berlin Heidelberg.
4. Leo Breiman, Jerome Friedman, Charles J. Stone, & R.A. Olshen. (1984). *Classification and Regression Trees*. CRC press.
5. Matthias Budde, Matthias Berning, Christopher Baumgärtner, Florian Kinn, Timo Kopf, Sven Ochs, Frederik Reiche, Till Riedel, & Michael Beigl. (2013). Point & control—interaction in smart environments: you only click twice. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication* (pp. 303-306). ACM.
6. Tsung-Hsiang Chang & Yang Li. (2011). Deep shot: a framework for migrating tasks across devices using mobile phone cameras. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 2163-2172). ACM.
7. David A. Cohn, Zoubin Ghahramani, & Michael I. Jordan. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research, 4*, 129-145.
8. Thomas M. Cover & Peter E. Hart. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21-27.
9. Adrian A. de Freitas, Michael Nebeling, Xiang "Anthony" Chen, Junrui Yang, Akshaye Ranithangam, & Anind K. Dey. (2016). Snap-To-It: a user-inspired platform for opportunistic device interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, (pp. 5909-5920).
10. Thomas Dietz, Gerald T. Gardner, Jonathan Gilligan, Paul C. Stern, & Michael P. Vandenbergh. (2009). Household actions can provide a behavioral wedge to rapidly reduce US carbon emissions. In *Proceedings of the National Academy of Sciences*, *106*(44), 18452-18456.
11. Robin A. Douthitt. (1989). An economic analysis of the demand for residential space heating fuel in Canada. *Energy*, *14*(4), 187-197.

12. Kirstin Early, Stephen E. Fienberg, & Jennifer Mankoff. (2016). Dynamic question ordering in online surveys. *arXiv preprint arXiv:* 1607.04209.
13. Denzil Ferreira, Eija Ferreira, Jorge Goncalves, Vassilis Kostakos, & Anind K. Dey. (2013). Revisiting human-battery interaction with an interactive battery interface. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 563-572). ACM.
14. Denzil Ferreira, Vassilis Kostakos, & Anind K. Dey. (2015). AWARE: Mobile context instrumentation framework. *Frontiers in ICT*, *2*(6), 1-9.
15. Pedro Ferreira, Pedro Sanches, Kristina Höök, & Tove Jaensson. (2008). License to chill! How to empower users to cope with stress. In *Proceedings of the Fifth Nordic Conference on Human-Computer Interaction: Building Bridges* (pp. 123-132). ACM.
16. David Ferrucci, Anthony Levas, Sugato Bagchi, David Gondek, & Erik T. Mueller. (2013). Watson: Beyond Jeopardy!. *Artificial Intelligence*, *199*, 93-105.
17. Jerome H. Friedman, Ron Kohavi, & Yeogirl Yun. (1996). Lazy decision trees. In *Proceedings of the 1st International Conference on Artificial Intelligence* (pp. 717-724). AAAI Press.
18. Nadav Golbandi, Yehuda Koren, & Ronny Lempel. (2011). Adaptive bootstrapping of recommender systems using decision trees. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining* (pp. 595-604). ACM.
19. Isabelle Guyon & André Elisseeff. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research, 3*, 1157-1182.
20. He He, Hal Daumé III, & Jason Eisner. (2012). Cost-sensitive dynamic feature selection. In *ICML Inferning Workshop*.
21. He He, Hal Daumé III, & Jason Eisner. (2013). Dynamic feature selection for dependency parsing. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1455-1464).
22. Paul A. Hirschberg, Elliot Abrams, Andrea Bleistein, William Bua, Luca Delle Monache, Thomas W. Dulong, John E. Gaynor, Bob Glahn, Thomas M. Hamill, James A. Hansen, Douglas C. Hilderbrand, Ross N. Hoffman, Betty Hearn Morrow, Brenda Philips, John Sokich, & Neil Stuart. (2011). A weather and climate enterprise strategic implementation plan for generating and communicating forecast uncertainty information. *Bulletin of the American Meteorological Society*, *92*(12), 1651-1666.
23. Suzanne S. Hudd, Jennifer Dumlao, Diane Erdmann-Sager, Daniel Murray, Emily Phan, Nicholas Soukas, & Nori Yokozuka. (2000). Stress at college: Effects on health habits, health status and self-esteem. *College Student Journal, 34*(2), 217– 227.
24. Madhu Kalia. (2002). Assessing the economic impact of stress—the modern-day hidden epidemic. *Metabolism, 51*(6), 49-53.
25. Nikhil Kaza. (2010). Understanding the spectrum of residential energy consumption: a quantile regression approach. *Energy Policy*, *38*(11), 6574-6585.
26. Blerina Lika, Kostas Kolomvatsos, & Stathes Hadjiefthymiades. (2014). Facing the cold start problem in recommender systems. *Expert Systems with Applications*, *41*(4), 2065-2073.
27. Qiong Liu, Paul McEvoy, Don Kimber, Patrick Chiu, & Hanningn Zhou. (2006). On redirecting documents with a mobile camera. In *8th IEEE Workshop on Multimedia Signal Processing* (pp. 467-470). IEEE.
28. David G. Lowe. (1999). Object recognition from local scale-invariant features. In *Proceedings of the 7th IEEE International Conference on Computer Vision* (pp. 1150-1157).
29. Hong Lu, Jun Yang, Zhigang Liu, Nicholas D. Lane, Tanzeem Choudhury, & Andrew T. Campbell. (2010). The Jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems* (pp. 71-84). ACM.
30. Chulhong Min, Chungkuk Yoo, Inseok Hwang, Seungwoo Kang, Youngki Lee, Seungchul Lee, Pillsoon Park, Changhun Lee, Seungpyo Choi, & Junehwa Song. (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). ACM.
31. Ranjita Misra & Michelle McKean. (2000). College students' academic stress and its relation to their anxiety, time management, and leisure satisfaction. *American Journal of Health Studies*, *16*(1), 41-51.
32. Gary P. Moberg. (2000). Biological response to stress: implications for animal welfare. In Gary P. Moberg & Joy A. Mench (Eds.), *The Biology of Animal Stress: Basic Principles and Implications for Animal Welfare* (1-21). New York: CABI Publishing.
33. Erman Pattuk, Murat Kantarcioglu, Huseyin Ulusoy, & Bradley Malin. (2015). Privacy-aware dynamic feature selection. In *31st IEEE International Conference on Data Engineering* (pp. 78-88). IEEE.
34. Nishkam Ravi, James Scott, & Liviu Iftode. (2008). Context-aware battery management for mobile phones. In *Sixth Annual IEEE International Conference on Pervasive Computing and Communications* (pp. 224-233). IEEE.

35. John Seryak & Kelly Kissock. (2003). Occupancy and behavioral effects on residential energy use. In *Proceedings of the Solar Conference* (pp. 717-722).

36. Kirill Trapeznikov & Venkatesh Saligrama. (2013). Supervised sequential classification under budget constraints. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics* (pp. 581-589).

37. Shannon E. Ross, Bradley C. Niebling, & Teresa M. Heckert. (1999). Sources of stress among college students. *College Student Journal, 33*(2), 312-317.

38. Maytal Saar-Tsechansky & Foster Provost. (2007). Handling missing values when applying classification models. *The Journal of Machine Learning Research, 8*, 1625-1657.

39. Mehdi Samadi, Partha Talukdar, Manuela Veloso, & Tom Mitchell. (2015). AskWorld: Budget-sensitive query evaluation for knowledge-on-demand. In *Proceedings of the 24th International Conference on Artificial Intelligence* (pp. 837-843). AAAI Press.

40. Nandita Sharma & Tom Gedeon. (2012). Objective measures, sensors and computational techniques for stress recognition and classification: A survey. *Computer Methods and Programs in Biomedicine*, *108*(3), 1287-1301.

41. Tianlin Shi, Jacob Steinhardt, & Percy Liang. (2015). Learning where to sample in structured prediction. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)* (pp. 875-884).

42. Emma Strubell, Luke Vilnis, Kate Silverstein, & Andrew McCallum. (2015). Learning dynamic feature selection for fast sequential prediction. *arXiv preprint arXiv:1505.06169*.

43. Mingxuan Sun, Fuxin Li, Joonseok Lee, Ke Zhou, Guy Lebanon, & Hongyuan Zha. (2013). Learning multiple-question decision trees for cold-start recommendation. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining* (pp. 445-454).

44. Lukas G. Swan & V. Ismet Ugursal. (2009). Modeling of end-use energy consumption in the residential sector: A review of modeling techniques. *Renewable and Sustainable Energy Reviews*, *13*(8), 1819-1835.

45. Joel A. Tropp. (2004). Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory, 50*(10), 2231-2242.

46. U.S. Energy Information Administration. (2009). *Residential Energy Consumption Survey 2009*. www.eia.gov/consumption/residential/data/2009/.

47. Rui Wang, Fanglin Chen, Zhenyu Chen, Tianxing Li, Gabriella Harari, Stefanie Tignor, Xia Zhou, Dror Ben-Zeev, & Andrew T. Campbell. (2014). StudentLife: Assessing mental health, academic performance, and behavioral trends of college students using smartphones. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 3-14).

48. Sanford Weisberg. (2014). *Applied Linear Regression* (4th ed.). New York: Wiley.

49. David J. Weiss & Ben Taskar. (2013). Learning adaptive value of information for structured prediction. In *Advances in Neural Information Processing Systems* (pp. 953-961).

50. Jiahui Wu, Gang Pan, Daqing Zhang, Shijian Li, & Zhaohui Wu. (2010). MagicPhone: pointing & interacting. In *Proceedings of the 2010 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication* (pp. 451-452). ACM.

51. Zhixiang (Eddie) Xu, Matt J. Kusner, Killian Q. Weinberger, Minmin Chen, & Olivier Chapelle. (2014). Classifier cascades and trees for minimizing feature evaluation cost. *The Journal of Machine Learning Research*, *15*(1), 2113-2144.