Feature Multi-Selection among Subjective Features

Sivan Sabato SIVAN.SABATO@MICROSOFT.COM
Adam Kalai ADAM.KALAI@MICROSOFT.COM
Microsoft Research New England, 1 Memorial Dr., Cambridge, MA

Abstract

When dealing with subjective, noisy, or otherwise nebulous features, the "wisdom of crowds" suggests that one may benefit from multiple judgments of the same feature on the same object. We give theoretically-motivated *feature multi-selection* algorithms that choose, among a large set of candidate features, not only which features to judge but how many times to judge each one. We demonstrate the effectiveness of this approach for linear regression on a crowd-sourced learning task of predicting people's height and weight from photos, using features such as *gender* and *estimated weight* as well as culturally fraught ones such as *attractive*.

1. Introduction

In this paper we consider prediction with subjective, vague, or noisy attributes (which are also termed 'features' throughout this paper). Such attributes can sometimes be useful for prediction, because they account for an important part of the signal that cannot be otherwise captured. In a crowdsourcing setting, the "wisdom of crowds" suggests that including multiple assessments of the same feature by different people may be useful. Henceforth, we refer to assessments of features as *judgments*. This paper introduces the problem of selecting, from a set of candidate features, which ones to use for prediction, and *how many judgments to acquire for each*, for a given budget limiting the total number of judgments. We give theoretically justified algorithms for this problem, and report crowd-sourced experimental results, in which judgments of highly subjective features (even culturally fraught ones such as *attractive*) are helpful for prediction.

As a toy example, consider the problem of estimating the number of jelly beans in a jar based on an image of the jar. A linear regressor with multiple judgments of features might have the form,

$$\hat{y} = 0.95(\text{est. number of beans})^{/5} - 50(\text{round jar})^{/2} + 100(\text{monochromatic})^{/1} + 30(\text{beautiful})^{/3}.$$

Here, for binary attributes, $a^{/r_a} \in [0,1]$ denotes the fraction of positive judgments out of r_a judgments of attribute a. For real-valued attributes, $a^{/r_a}$ denotes the mean of r_a judgments. The shape, number of colors, and attractiveness of the jar each help correct biases in the estimated number of beans, averaged across five people. Our goal is to choose a regressor that, as accurately as possible, estimates the labels (i.e., jelly bean counts) on future objects (i.e., jars) drawn from the same distribution, while staying within a budget of feature judgment resources per evaluated object at test time. In the example above, notice that even though the *monochromatic* coefficient is greater than the *beautiful* coefficient, fewer monochromatic judgments are used, because counting the number of colors is more objective, and hence further judgments are less valuable. While this example is contrived, similar phenomena are observed in the output of our algorithms.

We refer to the problem of selecting the number of repetitions, r_a, of each attribute, as the *feature multi-selection* problem, because it generalizes the feature selection problem of choosing a subset of features, i.e., $r_a \in \{0,1\}$, to choosing a multiset of features, i.e., $r_a \in \mathbb{N}$. Since the feature selection problem is well known to be NP-hard (Natarajan, 1995), our problem is also NP-hard in the general case. (For a formal reduction, one simply considers the "objective" case where all judgments of the same feature-object pair are identical.) Nonetheless, several successful approaches have been proposed for feature selection. The algorithms that we propose generalize two of these approaches to the problem of feature multi-selection.

Our algorithms are theoretically motivated, and tested on synthetic and real-world data. The real world data are photos extracted from the publicly available Photographic Height/Weight Chart (Cockerham, 2013), where people post pictures of themselves announcing their own height and weight.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

As a more general motivation, consider a scientist who would like to use crowdsourcing as an alternative to themselves estimating a value for each of a large data set of objects. Say the scientist gathers multiple judgments of a number of binary or real-valued attributes for each object, and uses linear regression to predict the value of interest. In some cases, crowdsourcing is a natural source of judgments, as a great number of them may be acquired on demand, rapidly, and at very low cost. We assume the scientist has access to the following information:

- A **labeled set** of objects $(o, y) \in O \times \mathcal{Y}$ (with no judgments), where O is a set of *objects* and $\mathcal{Y} \subseteq \mathbb{R}$ is a set of ground-truth *labels* drawn independently from a distribution $\mathcal{D}$.
- A **crowd**, which is a large pool of *workers*.
- A possibly large set of candidate **attributes** $\mathcal{A}$. For any attribute $a \in \mathcal{A}$ and object $o \in O$, the judgment of a random worker from the crowd may be queried at a cost.
- A **budget** B, limiting the number of attribute judgments to be used when evaluating the regressor on a new unseen object.

Our approach is as follows:

1. Collect $k \geq 2$ judgments for each candidate attribute in $\mathcal{A}$, for each object in the labeled set.
2. Based on this data and the budget, decide *how many judgments* of each attribute to use in the regressor.
3. Collect additional judgments (as needed) on the labeled set so that each attribute has the number of judgments specified in the previous step.
4. Find a linear predictor based on the average judgment of each feature.[1]

Step 4 can be accomplished by simple least-squares regression. The goal in Step 2 (feature multi-selection) is to decide on a number of judgments per attribute that will hopefully yield the smallest squared error after Step 4.

Interestingly, even given as few as $k = 2$ judgments per attribute, one can project an estimate of the squared error with more than k judgments of some features. We prove that these projections are accurate, for any fixed $k \geq 2$, as the number of labeled objects increases. Our algorithms perform a greedy strategy for feature multi-selection, to attempt to minimize the projected loss. This greedy strategy can be seen as a generalization of the Forward Regression approach for standard feature selection (see e.g. Miller, 2002). The first algorithm operates under the assumption that different attributes are uncorrelated. In this case the projection simplifies to a simple scoring rule, which incorporates attribute-label correlations as well as a natural notion of inter-rater reliability for each attribute. In this case, greedy selection is also provably optimal. While attributes are highly correlated in practice, the algorithm performs well in our experiments, possibly because Step 4 corrects for a small number of poor choices during feature multi-selection. The second algorithm attempts to optimize the projection without any assumptions on the nature of correlations between features.

While crowdsourcing is one motivation, the algorithms would be applicable to other settings such as learning from noisy sensor inputs, where one may place multiple sensors measuring each quantity, or social science experiments, where one may have multiple research assistants (rather than a crowd) judging each attribute.

The main contributions of this paper are: (a) introducing the feature multi-selection problem, (b) giving theoretically justified feature multi-selection algorithms, and (c) presenting experimental results, showing that feature multi-selection can yield more accurate regressors, with different numbers of judgments for different attributes. Proofs of results and additional experimental data are provided in Sabato & Kalai (2013).

Related Work

Related work spans a number of fields, including Statistics, Machine Learning, Crowdsourcing, and measurement in the social sciences. A number of researchers have studied *attribute-efficient prediction* (also called *budgeted learning*) assuming, as we do, that there is a cost to evaluating attributes and one would like to evaluate as few as possible (see, for instance, the recent work by Cesa-Bianchi et al. (2011) and references therein). In that line of work, each attribute is judged at most once. The errors-in-variables approach (e.g., Cheng & Van Ness, 1999) in statistics estimates the 'true' regression coefficients using noisy feature measurements. This approach is less suitable in our setting, since our final goal is to predict from noisy measurements.

A wide variety of techniques have been studied to com-

[1]We focus on mean averaging, leaving to future work other aggregation statistics such as the median.

bine estimates of experts or the crowd of a single quantity of interest (see, e.g. Dawid & Skene, 1979; Smyth et al., 1994; Welinder et al., 2010), like estimating the number of jelly beans in a jar from a number of guesses.

Two recent works on crowdsourcing are very relevant. Patterson & Hays (2012) crowdsourced the mean of 3 judgments of each of 102 binary attributes on over 14,000 images, yielding over 4 million judgments. Some of their attributes are subjective, e.g., *soothing*. We employ their crowdsourcing protocol to label our binary attributes. Isola et al. (2011) study subjective and objective features for the task of estimating how memorable an image is, by taking the mean of 10 judgments per attribute for each image. They perform greedy feature selection over these attributes to find the best compact set of attributes for predicting memorability. The key difference between their algorithm and ours is that theirs does not choose how many judgments to average. Since that quantity is fixed for each attribute, their setting falls under the more standard feature selection umbrella. In our experiments we compare this approach to our algorithms.

Finally, in the social sciences, a wide array of techniques have been developed for assessing inter-rater reliability of attributes, with the most popular perhaps being the α coefficient (Cronbach, 1951). A principal use of such measures is determining, by some threshold, which features may be used in content analysis. For an overview of reliability theory, see (Krippendorff, 2012).

2. Preliminary assumptions and definitions

Let there be d candidate attributes called $\mathcal{A} = [d] = \{1, 2, \ldots, d\}$. We assume that, for any object o and attribute a, there is a distribution over judgments $\mathbb{P}[X[a] \mid O = o]$, and we assume that the judgments of attribute-object pairs are conditionally independent given the sets of attributes and objects. This represents an idealized setting in which a new random crowd worker is selected for each attribute-object judgment (In our experiments, we limit the total amount of work that any one worker may perform). We assume a distribution $\mathcal{D}$ over labeled objects, where labels are real numbers. We denote by $\mathcal{D}_O$ the marginal distribution over objects drawn according to $\mathcal{D}$. We let $\mathbb{P}[X[a]] = \mathbb{P}_{O\sim\mathcal{D}_O}[X[a] \mid O]]$. Labels y are assumed to be real valued. As is standard, we assume one "true" label y_i for each object o_i.

For notational ease, we assume that in the feature multi-selection phase, exactly $k \geq 2$ judgments for each feature are collected. Our analysis trivially generalizes to the setting in which different attributes are judged different numbers of times. Finally, each attribute a is assumed to have an expected value of $\mathbb{E}[X[a]] = 0$, where the expectation is taken across objects and judgments of a. This is done for ease of presentation, so that we do not have to track the mean vectors as well as the variance. When discussing implementation details, we describe how to remove this assumption in practice without loss of generality.

Vectors will be boldface, e.g., $\mathbf{x} = (x[1], \ldots, x[d])$, random variables will be capitalized, e.g., X, and matrices will be in black-board font, e.g., $\mathbb{X}$. The i'th standard unit vector is denoted by $\mathbf{e}_i$.

Let $\mathbf{r} \in \mathbb{N}^d$ represent the number of judgments for each feature, so that attribute a is judged $r[a]$ times, and we represent the object's judgments by $\mathbf{x}$, defined as:

$$\mathbf{x} = \left(\langle x[1](j)\rangle_{j=1}^{r[1]}, \ldots, \langle x[d](j)\rangle_{j=1}^{r[d]}\right),$$

where $x[a](j)$ is the jth judgment of attribute a in $\mathbf{x}$, and $\langle x[a](j)\rangle_{j=1}^{r[a]}$ is a vector with $x[a](j)$ in coordinate j. We say that $\mathbf{r}$ is the *repeat vector* of $\mathbf{x}$. We denote the set of all possible representations with repeat vector $\mathbf{r}$ by $\mathbb{R}^{[\mathbf{r}]}$.

We denote by $D_\mathbf{r}$ the distribution which draws $(\mathbf{X}, Y) \in \mathbb{R}^{[\mathbf{r}]} \times \mathbb{R}$ by first drawing a labeled object (O, Y) from $\mathcal{D}$, and then drawing a random representation $\mathbf{X} \in \mathbb{R}^{[\mathbf{r}]}$ for this object. We denote by D_∞ the distribution that draws $(\mathbf{X}, Y)$ where $\mathbf{X} \in \mathbb{R}^d$ by first drawing (O, Y) from $\mathcal{D}$ and then setting $X[a] = \mathbb{E}[X[a] \mid O]$. We denote the expectation over $D_\mathbf{r}$ by $\mathbb{E}_\mathbf{r} = \mathbb{E}_{(\mathbf{X},Y)\sim D_\mathbf{r}}$. For D_∞ we denote $\mathbb{E}_\infty = \mathbb{E}_{(\mathbf{X},Y)\sim D_\infty}$.

For $k \geq 2$, let $\mathbf{k} = (k, k, \ldots, k) \in \mathbb{N}^d$ be the repeat vector used in the first training phase. The feature multi-selection algorithm receives as input a labeled training set $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$ where $\mathbf{x}_i \in \mathbb{R}^{kd}$ and $y_i \in \mathbb{R}$, drawn from $D_\mathbf{k}$. This sample is generated by first drawing a set of labeled objects $((o_1, y_1), \ldots, (o_m, y_m))$ i.i.d. from $\mathcal{D}$, and then drawing a random representation $\mathbf{x}_i$ for object o_i. The algorithm further receives as input a budget $B \in \mathbb{N}$, which specifies the total number of feature judgments allowed for each unlabeled object at test (i.e., prediction) time. The output of the algorithm is a new vector of repeats $\mathbf{r} \in R_B$, where,

$$R_B \equiv \left\{\mathbf{r} \in \mathbb{N}^d \mid \sum\nolimits_{a\in\mathcal{A}} r[a] \leq B\right\}.$$

Let o be an object with a true label y, and let $\hat{y}$ be a prediction of the label of o. The squared loss for this prediction is $\ell(y, \hat{y}) = (y - \hat{y})^2$. Given a function $f : Z \to \mathbb{R}$ for some domain Z, and a distribution D over $Z \times \mathbb{R}$, we denote the average loss of f on D by

$$\ell(f, D) \equiv \mathbb{E}_{(Z,Y)\sim D}[\ell(f(Z), Y)].$$

The final goal of our procedure is to find a predictor with a low expected loss on labeled objects drawn from $\mathcal{D}$. This predictor must use only B feature judgments for each object, as determined by the test repeat vector $\mathbf{r}$. We consider linear predictors $\mathbf{w} \in \mathbb{R}^d$ that operate on the vector of *average* judgments of $\mathbf{x} \in \mathbb{R}^{[\mathbf{r}]}$, defined as follows:

$$\bar{x}[a] \equiv \begin{cases} \frac{1}{r[a]} \sum_{j=1}^{r[a]} x[a](j) & \text{if } r[a] > 0, \\ 0 & \text{if } r[a] = 0. \end{cases}$$

For an input representation $\mathbf{x}$, the predictor $\mathbf{w}$ predicts the label $\langle \mathbf{w}, \bar{\mathbf{x}} \rangle$. For vector $\mathbf{v} \in \mathbb{R}^d$, we denote by $\mathrm{Diag}(\mathbf{v}) \in \mathbb{R}^{d\times d}$ the diagonal matrix with $v[a]$ in the ath position.

For a vector $\mathbf{r} \in \mathbb{N}^d$ and a matrix $\mathbb{S} \in \mathbb{R}^{d\times d}$, we denote by $\mathrm{sub}_{\mathbf{r}}(\mathbb{S})$ the submatrix of $\mathbb{S}$ resulting from deleting all rows and columns a such that $r[a] = 0$. For a vector, $\mathrm{sub}_{\mathbf{r}}(\mathbf{u})$ omits entries a such that $r[a] = 0$. Here $\mathrm{sub}_{\mathbf{r}}(\mathbf{u}) \in \mathbb{R}^{d'}$ and $\mathrm{sub}_{\mathbf{r}}(\mathbb{S}) \in \mathbb{R}^{d'\times d'}$, where d' is the support size of $\mathbf{r}$. We denote the pseudo-inverse of a matrix $\mathbb{A} \in \mathbb{R}^{n\times n}$ (see e.g. Ben-Israel & Greville, 2003) by $\mathbb{A}^+$.

3. Feature Multi-Selection Algorithms

The input to a feature multi-selection algorithm is a budget B and m labeled examples in which each attribute has been judged k times, and the output is a repeat vector $\mathbf{r} \in R_B$. Our ultimate goal is to find $\mathbf{r}$ and a predictor $\mathbf{w} \in \mathbb{R}^d$ such that $\ell(\mathbf{w}, D_{\mathbf{r}})$ is minimal. We now give intuition about the derivation of the algorithms, but their formal definition is given in Alg. 1.

Define the loss of a repeat vector to be $\ell(\mathbf{r}) \equiv \min_{\mathbf{w}\in\mathbb{R}^d} \ell(\mathbf{w}, D_{\mathbf{r}})$. The goal is to minimize $\ell(\mathbf{r})$ over $\mathbf{r} \in R_B$. We give two forward-selection algorithms, both of which begin with $\mathbf{r} = (0, \ldots, 0)$ and greedily increment $r[a]$ for a that most decreases an estimate of $\ell(\mathbf{r})$. The key question is how does one estimate this projected loss $\ell(\mathbf{r})$ since the number of judgments can exceed k. We simplify notation by first considering only $\mathbf{r}$ which are *positive*, i.e., $r[a] \geq 1$ for each a. We will shortly explain how to handle $r[a] = 0$. Define

$$\mathbf{b} = \mathbb{E}[\mathbf{X}Y], \text{ and } \Sigma_{\mathbf{r}} = \mathbb{E}_{\mathbf{r}}[\bar{\mathbf{X}}^T\bar{\mathbf{X}}].$$

We call $b[a]$ the *correlation* of a with the label. Note that $\mathbf{b} = \mathbb{E}_{\mathbf{k}}[\bar{\mathbf{X}}Y]$, since linearity of expectation implies that $\mathbf{b}$ does not depend on $\mathbf{k}$. Straightforward calculations show that, for any positive repeat vector $\mathbf{r}$, If $\Sigma_{\mathbf{r}}$ is non-singular,[2]

$$\ell(\mathbf{r}) = \min_{\mathbf{w}} \mathbb{E}_{\mathbf{r}}\left[(\mathbf{w}^T\bar{\mathbf{X}} - Y)^2\right] = \mathbb{E}_{\mathbf{r}}[Y^2] - \mathbf{b}^T\Sigma_{\mathbf{r}}^{-1}\mathbf{b}.$$

Since $\mathbb{E}[Y^2]$ does not depend on $\mathbf{r}$, minimizing $\ell(\mathbf{r})$ is equivalent to maximizing $\mathbf{b}^T\Sigma_{\mathbf{r}}^{-1}\mathbf{b}$ (for positive $\mathbf{r}$ and nonsingular $\Sigma_{\mathbf{r}}$).

3.1. A Scoring Algorithm

The first algorithm that we propose is derived from the zero-correlation assumption, that $\mathbb{E}[X[a]X[a']] = 0$ for $a \neq a'$, or equivalently that the covariance matrix is diagonal. Perhaps the simplest approach to standard feature selection is to score each feature independently, based on its normalized empirical correlation with the label, and to select the B top-scoring features. If features are uncorrelated and the training sample is sufficiently large, then this efficient approach finds an optimal set of features. The feature multi-selection scoring algorithm that we propose henceforth is optimal under similar assumptions, however it is complicated by the fact that we may include multiple repetitions of each feature. Under the zero-correlation assumption, $\Sigma_{\mathbf{r}}$ is diagonal, and its ath element, for $r[a] > 0$, can be expanded as

$$\begin{aligned} \mathbb{E}_{\mathbf{r}}[(\bar{X}[a])^2] &= \sigma^2[a] + \frac{v[a]}{r[a]}, \text{ where} \\ v[a] &\equiv \mathbb{E}_{O\sim\mathcal{D}_O}[\mathrm{Var}[X[a] \mid O]] \text{ and} \\ \sigma^2[a] &\equiv \mathbb{E}_\infty\left[(X[a])^2\right]. \end{aligned}$$

We refer to $v[a]$ as the *internal variance* as it measures the "inter-rater reliability" of a, and we call $\sigma^2[a]$ the *external variance* as it is the inherent variance between examples. Hence for a diagonal $\Sigma_{\mathbf{r}}$, simple manipulation gives,

$$\mathbb{E}[Y^2] - \ell(\mathbf{r}) = \sum_{a:r[a]>0} \frac{(b[a])^2}{\sigma^2[a] + \frac{v[a]}{r[a]}}. \tag{1}$$

Therefore, when $\Sigma_{\mathbf{r}}$ is diagonal, minimizing the projected loss is equivalent to maximizing the RHS above, a sum of independent terms that depend on the correlation and on the internal and external variance of each attribute, all of which can be estimated just once, for all possible repeat vectors. As one expects, greater correlation indicates a better feature, while a greater external variance indicates a worse feature. A larger internal variance indicates that more repeats are needed to achieve prediction quality.

[2]For singular $\Sigma_{\mathbf{r}}$, the pseudo-inverse $\Sigma_{\mathbf{r}}^+$ replaces $\Sigma_{\mathbf{r}}^{-1}$.

To estimate Eq. (1) we estimate each of the components on the RHS. Unbiased estimation of $\mathbf{b}$ is straightforward, and unbiased estimation of $\mathbf{v}$ is also possible for $k \geq 2$ samples per object, though importantly one should use the unbiased variance estimator,

$$\hat{v}[a] = \frac{1}{m}\sum_i \mathrm{VarEst}(x_i[a](1), \ldots, x_i[a](j)), \tag{2}$$

$$\mathrm{VarEst}(\alpha_1, \ldots, \alpha_n) \equiv \frac{1}{n-1}\sum_{j\in[n]}(\alpha_j - \frac{1}{n}\sum_{j'\in[n]}\alpha_{j'})^2.$$

Using these estimates of $\mathbf{v}$, we estimate the external variance using the equality $\sigma^2[a] = \mathbb{E}_{\mathbf{k}}\left[(\bar{X}[a])^2\right] - \frac{v[a]}{k}$. A slight complication arises here, as this estimate might be negative for small samples, so we round it up to 0 when this happens. Another issue might seem to arise when the denominator of one of the summands in Eq. (1) is zero, however note that this can only occur if both the internal and the external variance are zero, which implies that the feature is constantly zero, thus zeroing its correlation as well. The same holds for the estimated ratio. In such cases we treat the ratio as equal to 0.

Algorithm 1 Feature multi-selection algorithms

```
1: Input: Budget $B$; $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in \mathbb{R}^{dk+1}$, Algorithm type: Scoring/Full.
2: Output: A repeat vector $\mathbf{r} \in R_B$.
3: $\bar{x}_i[a] \leftarrow \frac{1}{k}\sum_{j\in[k]} x_i[a](j)$ for $i \in [m], a \in A$.
4: $\hat{\mathbf{b}} \leftarrow \frac{1}{m}\sum_i y_i \bar{\mathbf{x}}_i$.
5: $\hat{v}[a] \leftarrow \frac{1}{m}\sum_i \mathrm{VarEst}(x_i[a](1), \ldots, x_i[a](k))$.
6: if Scoring Algorithm then
7:     $\forall a \in A, \hat{\sigma}^2[a] \leftarrow \max\left\{0, \frac{1}{m}\sum_i(\bar{x}_i[a])^2 - \frac{\hat{v}[a]}{k}\right\}$.
8:     Define $\hat{\mathrm{obj}}(\mathbf{r}) \equiv \sum_{a:r[a]>0} \hat{b}[a]^2/(\hat{\sigma}^2[a] + \frac{\hat{v}[a]}{r[a]})$
9: else
10:    $\hat{\Sigma} \leftarrow \mathrm{MakePSD}\left(\frac{1}{m}\sum_i \bar{\mathbf{x}}_i^T\bar{\mathbf{x}}_i - \mathrm{Diag}(\hat{\mathbf{v}})/k\right)$
11:    $\mathbb{M}_{\mathbf{r}} \equiv \mathrm{sub}_{\mathbf{r}}(\hat{\Sigma} + \mathrm{Diag}(\frac{\hat{v}[1]}{r[1]}, \ldots, \frac{\hat{v}[d]}{r[d]}))$
12:    Define $\hat{\mathrm{obj}}(\mathbf{r}) \equiv \mathrm{sub}_{\mathbf{r}}(\hat{\mathbf{b}})^T\mathbb{M}_{\mathbf{r}}^{+}\mathrm{sub}_{\mathbf{r}}(\hat{\mathbf{b}})$
13: end if
14: $\mathbf{r}_0 \leftarrow (0, \ldots, 0) \in \mathbb{N}^d$
15: for $t = 1$ to $B$ do
16:    Find $i_{\mathrm{best}} \in [d]$ such that $\hat{\mathrm{obj}}(\mathbf{r}_{t-1} + \mathbf{e}_i)$ is maximal.
17:    $\mathbf{r}_t \leftarrow \mathbf{r}_{t-1} + \mathbf{e}_{i_{\mathrm{best}}}$.
18: end for
19: Return $\mathbf{r}_B$.
```

3.2. The Full Multi-Selection Algorithm

The scoring algorithm is motivated by the assumption of zero correlation between features. However, this assumption rarely holds in practice. Building on and paralleling the definitions and derivation above, the *Full Algorithm* similarly maximizes $\mathbf{b}^T\Sigma_{\mathbf{r}}^{-1}\mathbf{b}$ without this assumption. For positive $\mathbf{r}$, one has

$$\Sigma_{\mathbf{r}} = \Sigma + \mathrm{Diag}(v[1]/r[1], \ldots, v[d]/r[d])$$

Where $\Sigma \equiv \mathbb{E}_\infty[\mathbf{X}^T\mathbf{X}]$ is the *external covariance* matrix, and we estimate it based on the equality $\Sigma = \Sigma_{\mathbf{k}} - \mathrm{Diag}(\mathbf{v})/k$. Just as in the Scoring algorithm, the estimates of $\sigma^2[a]$ might be negative, in the full algorithm it is possible that the estimate of Σ will not be positive semi-definite, so we analogously "round up" our estimate of Σ to the nearest PSD matrix (see implementation details below). The estimate when some of the $r[a]$'s are zero is formed by deleting the corresponding entries in the estimate of $\mathbf{b}$ and the corresponding rows and columns in the estimate of $\Sigma_{\mathbf{r}}$.

3.3. Guarantees

Under our distributional assumptions, we show that the estimated objective functions used by our algorithms converge to $\mathbb{E}[Y^2] - \ell(\mathbf{r})$. Thus maximizing the estimated objective approximately minimizes $\ell(\mathbf{r})$. Formally, let $\hat{\mathrm{obj}}_f(\mathbf{r})$ and $\hat{\mathrm{obj}}_s(\mathbf{r})$ be the objectives used in Alg. 1 for the full algorithm and the Scoring algorithm, respectively. Note that these objectives are implicitly functions of the training sample S. For a symmetric matrix $\mathbb{S}$, let $\lambda_{\min}(\mathbb{S})$ be the smallest eigenvalues of $\mathbb{S}$. We define: $\lambda = \min_{\mathbf{r}\in R_B}\lambda_{\min}(\mathrm{sub}_{\mathbf{r}}(\Sigma))$, and $\bar{B} = \min(B, d)$.

Theorem 3.1. *Suppose that all judgments and labels are in $[-1, 1]$. Then for any $\delta \in (0, 1)$, with prob. at least $1 - \delta$ over m i.i.d. training samples from $D_{\mathbf{k}}$, for all $\mathbf{r} \in R_B$, for $m \geq \tilde{\Omega}(\bar{B}\ln(\bar{B}d/\delta)/\lambda^2)$ we have*

$$|\hat{\mathrm{obj}}_f(\mathbf{r}) - (\mathbb{E}[Y^2] - \ell(\mathbf{r}))| \leq O\left(\frac{\bar{B}^3\ln(Bd/\delta)}{\lambda^2\sqrt{m}}\right).$$

If the external covariance matrix Σ is diagonal, then for $m \geq \tilde{\Omega}(\ln(d/\delta)/\lambda^2)$ we have

$$|\hat{\mathrm{obj}}_s(\mathbf{r}) - (\mathbb{E}[Y^2] - \ell(\mathbf{r}))| \leq O\left(\frac{\ln(Bd/\delta)}{\lambda^2\sqrt{m}}\right).$$

The convergence rate for the full algorithm stems from two bounds: (1) If the norm of the minimizing $\mathbf{w}$ is at most α, then the convergence rate is at most $\bar{B}\alpha^2/\sqrt{m}$; (2) With high probability, the norm of the minimizing $\mathbf{w}$ is at most $\sqrt{\bar{B}}/\lambda$. An additional factor of $O(\bar{B}\ln(Bd))$ gets uniform convergence over $\mathbf{r} \in R_B$. The components of this result are of the same order as the equivalent results for uniform convergence of standard least-squares regression. An improved rate of

$\sqrt{\bar{B}\alpha^2/m}$ can be achieved for least-squares regression, *if the algorithm exactly minimizes the sample squared loss* (Srebro et al., 2010). However, our algorithm minimizes another objective, thus this result is not directly applicable. We leave it as a challenge for future work to find out whether a faster rate can be achieved in our case.

As always, these convergence rates are worst-case, and in practice a much smaller sample size is often sufficient to get meaningful results, as we have observed in our experiments. However, if the available training sample is too small to achieve reasonable results, one can limit the norm of the minimizer by adding regularization to the estimated covariance matrix, as in ridge regression (Hoerl & Kennard, 1970). This would allow faster convergence at the expense of a more limited class of predictors.

As Theorem 3.1 shows, when the zero-correlation assumption holds, the Scoring algorithm enjoys a much faster worst-case rate of convergence than the full algorithm. This is because it does not attempt to estimate the entire covariance matrix. This advantage is more significant for larger budgets. An additional advantage is that it finds the *optimal* value of $\mathbf{r}$ for its estimated objective:

Theorem 3.2. *The Scoring algorithm returns* $\mathbf{r} \in \mathrm{argmax}_{\mathbf{r} \in R_B} \hat{\mathrm{obj}}_s(\mathbf{r})$.

Theorem 3.2 follows since $f(r) = a/(b+c/r)$ is concave and increasing in r and from the following observation.

Lemma 3.3. *Let* $\mathbf{r} \in \mathbb{N}^d$, *and* $f(\mathbf{r}) = \sum_{i \in [d]} g_i(r[i])$, *where* $g_i(\cdot) : \mathbb{R}_+ \to \mathbb{R}$ *are monotonic non-decreasing concave functions. Let* $B \in \mathbb{N}$. *The maximum of* $f(\mathbf{r})$ *subject to* $\mathbf{r} \in R_B$ *is attained by a greedy algorithm which starts with* $\mathbf{r} = (0, \ldots, 0)$, *and iteratively increases the coordinate which increases* f *the most.*

3.4. Implementation

If our estimate of Σ is not PSD, we use the procedure 'MakePSD', which takes a symmetric matrix $\mathbb{A}$ as input, and returns the PSD closest to $\mathbb{A}$ in Frobenius norm. This can be done by calculating the eigenvalue decomposition $\mathbb{A} = \mathbb{U}\mathbb{D}\mathbb{U}^T$ where $\mathbb{U}$ is orthogonal and $\mathbb{D}$ is diagonal, and returning $\mathbb{U}\tilde{\mathbb{D}}\mathbb{U}^T$, where $\tilde{\mathbb{D}}$ is $\mathbb{D}$ with zeroed negative entries (Higham, 1988). If we assume a diagonal external covariance, then this procedure is equivalent to rounding up the estimate of $\sigma^2(a)$ to zero, as done in the Scoring algorithm. For a budget of B, the full algorithm performs Bd SVDs to calculate pseudo-inverses. Note, however, that the largest matrix that might be decomposed here is of size $\min(d, B) \times \min(d, B)$. Furthermore, in practice the matrices can be much smaller, since the algorithm might choose several repeats of the same features. In our experiments, the total time for decompositions, using standard libraries on a standard personal computer, has been negligible.

Our description of the algorithms above assumes for simplicity that the mean of all features is zero. In practice, one adds a 'free' feature that is always 1, to allow for biased regressors. For the Scoring algorithm, one should further subtract the empirical mean from each feature. For the full algorithm, this not necessary, because when bias is allowed, adding a constant to any feature provably will not change the output of the full algorithm.

4. Experiments

We tested our approach on three regression problems. In the first problem the feature judgments were simulated. In the second and third problem they were collected from the crowd using Amazon's Mechanical Turk.[3] For the simulated experiment we used the UCI dataset 'Relative location of CT slices on axial axis Data Set' (Frank & Asuncion, 2010). Here the features are histograms of spatial measurements in the image, and the label to predict is the relative location of the image on the axial axis. To simulate features with varying judgments, we collapsed each set of 8 adjacent histogram bins into a single feature, so that each judgment of the new feature was randomly chosen out of 8 possible values for this feature. The resulting dataset contained 48 noisy real-valued features per example.

The second and third problems were to predict the

[3]`http://mturk.com`. We will share our data upon request from other researchers, due to the sensitivity of judgments on people's images.

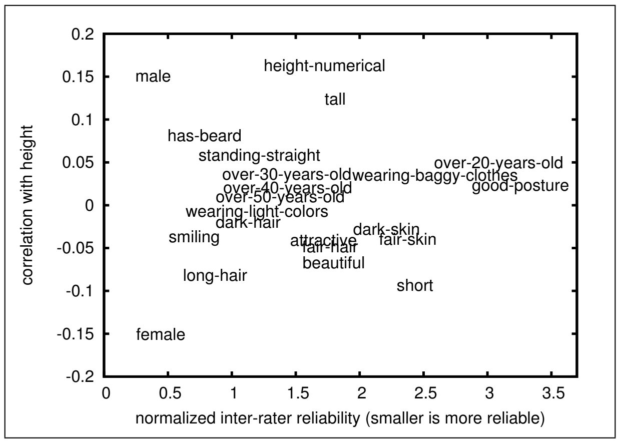


Figure 1. Properties of selected attributes for height

height and weight of people from a photo. 880 photos with self-declared height and weight were extracted from the publicly available Photographic Height/Weight Chart (Cockerham, 2013), where people post pictures of themselves announcing their own height and weight. We chose 37 attributes that we felt the crowd could judge and might be predictive. We collected judgments for these binary attributes, mainly following the collection methodology of Patterson & Hays (2012), by batching the images into groups of 40, to make judging efficient. To encourage honest workers, we promised (and delivered) bonuses for good work. We limited the amount of work any one person could do. We used all of the collected judgments, regardless of whether the workers received bonuses for them or not. Our pay per hour was set to average to minimum wage. We collected numerical estimates of the height and the weight in a similar fashion. Binary judgments took about one second per judgment and their cost was a fraction of a cent per attribute judgment. The numerical estimates took about four times as long and we paid four times as much for them. Accordingly, we adjusted all the algorithms to count a single numerical judgment as equal to four binary attribute judgments. Figure 1 and Figure 2 show the normalized correlation ($\hat{b}[a]/\hat{\sigma}[a]$) vs. the normalized inter-rater reliability ($\hat{v}[a]/\hat{\sigma}[a]$) of selected attributes. These plots demonstrate that all combinations of useful/non-useful and stable/noisy attributes exist in this data. The full listing of attribute properties is provided in (Sabato & Kalai, 2013).

We compared the test error of our algorithms, denoted 'Full' and 'Scoring' in the plots, to those of several plausible baselines. In all comparisons, we set $k = 2$. The first baseline, denoted 'Averages' in the plots, is based on the "predictive" feature selection algorithm of Isola et al. (2011): We average the 2 judgments per attribute to create a standard data set with one value for each object-attribute pair, and greedily add attributes, one at a time, so as to minimize the least-squares error. The resulting regressor uses 2 judgments for each selected feature. The second baseline, denoted 'Copies', treats the 2 judgments of each feature-object pair as 2 different individual attributes, and again performs greedy forward selection on these features, except that the order of selecting the copies in each feature was fixed, to avoid over-fitting. The test repeat vector $\mathbf{r}$ was set according to the number of copies selected for each feature. These baselines perform standard Machine Learning feature selection: 'Averages' considers d features and 'Copies' considers $2d$ features. For height and weight prediction, we compared the results also to the test error achieved by averaging only the height or weight estimates of the crowd, respectively. Since each numerical feature costs 4 times as much as a binary feature, we averaged over $B/4$ numerical judgments when the budget was set to B. We did not use regularization anywhere, thus our algorithms and the baselines are all parameter-free.

The test error in the plots was obtained as follows: $\mathbf{r}$ was selected based on a training set with $\mathbf{k}$ judgments. We then added judgments to features in the training set to get to $\mathbf{r}$ repeats. Finally we performed regular regression on the means of the enhanced training set to get a predictor. This predictor was used to predict the labels of the test set with $\mathbf{r}$ judgments. We averaged results over 50 random train/test splits. Figures 3 and 4 show that our full algorithm achieved better test error than the baselines. The Scoring algorithm was usually somewhat worse than the Full Multi-Selection algorithm, and for small budgets also sometimes worse than the baselines. This is expected due to its zero-correlation assumption. However, when the sample size was small, the Scoring algorithm was sometimes better (see third plot in Fig. 4), since it suffered from less over-fitting. This is consistent with our convergence analysis in Theorem 3.1. Analysis of training errors indicates that baseline algorithms suffer for two different reasons: (1) they are limited to a small number of repeats per feature; and (2) they suffer from greater over-fitting, probably since our algorithm tends to select a sparser $\mathbf{r}$ than do the baselines. We list examples of predictors learned by the full algorithm in (Sabato & Kalai, 2013).

In our last experiment we tested the tradeoff between no. of training judgments per feature and no. of training examples, in the following setting: Suppose we have a budget that allows us to collect a total of M judgments for training the feature multi-selection algorithm, and we have access to at least $M/2d$ labeled

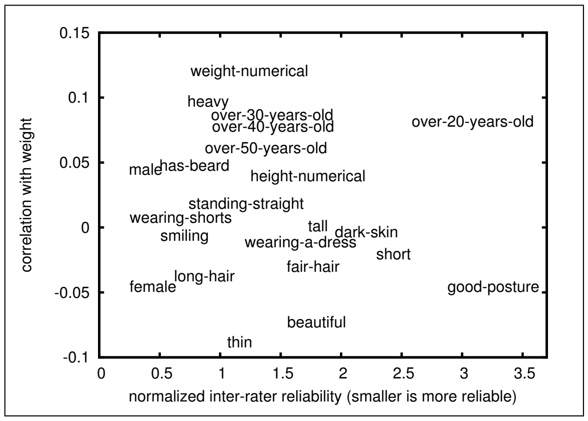
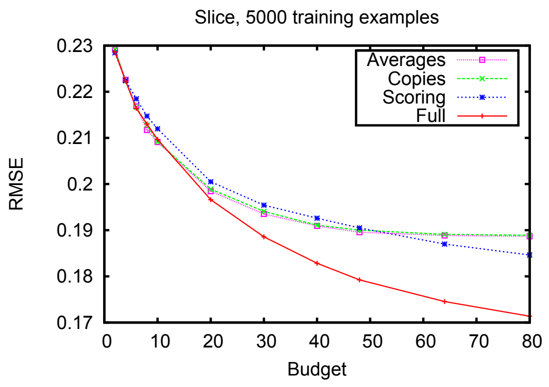


Figure 2. Properties of selected attributes for weight



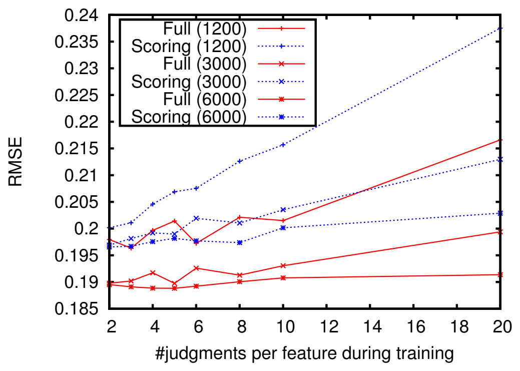


Figure 3. The 'Slice' dataset. Top: comparing algorithms. Bottom: Training loss with a constant #training examples×#repeats. Numbers in legend indicate value of constant. Budget was fixed to 30.

examples. We can decide on a number k of judgments per feature, randomly select M/kd objects from our labeled pool to serve as the training set, and obtain kd judgments for each of these objects. What number k should we choose? Does this number depend on the total budget M? We compared the test error arising from different values of k over different values of M, for the slice dataset using both of our algorithms,. The results are shown in Figure 3. These results show a clear preference for a small k (which allows a large m on the same budget M). Characterizing the optimal number k is left as an open question for future work.

5. Conclusions

We introduce the problem of *feature multi-selection* and provide two algorithms for regression with mean averaging of judgments. Future directions include other learning tasks, such as classification, and other types of feature aggregation, such as median averaging (majority). An additional important question for future work is how to carry out feature multi-selection in an environment with a changing crowd.

Acknowledgments. We wish to thank Edith Law and Haoqi Zhang for several helpful discussions.

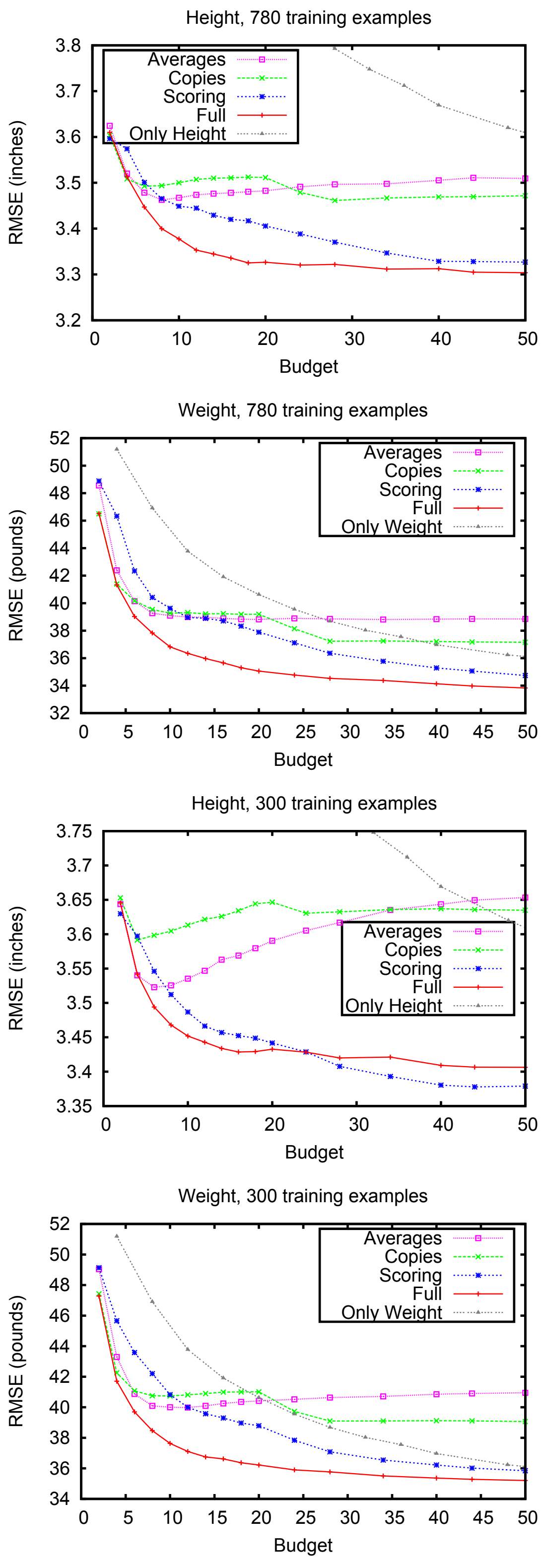


Figure 4. Comparisons for height and weight prediction.

References

Ben-Israel, A. and Greville, T.N.E. *Generalized inverses: theory and applications*, volume 15. Springer, 2003.

Cesa-Bianchi, N., Shalev-Shwartz, S., and Shamir, O. Efficient learning with partially observed attributes. *J. Mach. Learn. Res.*, 12:2857–2878, November 2011.

Cheng, C. and Van Ness, J.W. *Statistical regression with measurement error*, volume 6. Arnold London, 1999.

Cockerham, R. The photographic height and weight chart. http://www.cockeyed.com/photos/bodies/heightweight.html, 2013. With permission of Rob Cockerham.

Cronbach, L. Coefficient alpha and the internal structure of tests. *Psychometrika*, 16(3):297–334, 1951.

Dawid, A. P. and Skene, A. M. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):pp. 20–28, 1979.

Frank, A. and Asuncion, A. UCI machine learning repository, 2010. URL http://archive.ics.uci.edu/ml.

Higham, N. J. Computing a nearest symmetric positive semidefinite matrix. *Linear algebra and its applications*, 103:103–118, 1988.

Hoerl, A. E. and Kennard, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Isola, P., Parikh, D., Torralba, A., and Oliva, A. Understanding the intrinsic memorability of images. In *Advances in Neural Information Processing Systems 24*, pp. 2429–2437, 2011.

Krippendorff, K. *Content analysis: An introduction to its methodology*. Sage Publications, Incorporated, 2012.

Miller, A. *Subset selection in regression*. Chapman & Hall/CRC, 2002.

Natarajan, B.K. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2): 227–234, 1995.

Patterson, G. and Hays, J. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *Proceeding of the 25th Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Sabato, S. and Kalai, A. Feature multi-selection among subjective features. *CoRR*, abs/1302.4297, 2013. URL http://arxiv.org/abs/1302.4297.

Smyth, P., Fayyad, U. M., Burl, M. C., Perona, P., and Baldi, P. Inferring ground truth from subjective labelling of venus images. In *NIPS*, pp. 1085–1092, 1994.

Srebro, N., Sridharan, K., and Tewari, A. Smoothness, low-noise and fast rates. In *Advances in Neural Information Processing Systems (NIPS) 23*, 2010.

Welinder, P., Branson, S., Belongie, S., and Perona, P. The multidimensional wisdom of crowds. In *Neural Information Processing Systems Conference (NIPS)*, 2010.