

APPENDIX FOR LEVERAGING HUMAN FEATURES AT TEST-TIME

Anonymous authors

Paper under double-blind review

1 TOY EXAMPLE

We demonstrate the entropy selection approach on a pedagogical toy example where important human features with high uncertainty can be predicted from the machine features. If the important human features were fully predictable from the machine features, they would be unnecessary for classification, and if they were not predictable at all, our approach would not work.

1.1 DATA GENERATION

We generate a dataset with 10 human features, 20 machine features, and 5000 instances where each of the human features has a corresponding, unique machine feature that, when activated, corresponds to the value of the human feature being highly uncertain. In all other cases, the human features are highly unlikely to be activated for an instance. Exactly one of these machine features is activated for each instance. The remaining 10 machine features are uncorrelated with the human features, and are activated with $p = 0.5$. Labels are then sampled from a linear regression model with randomly generated weights where the paired human and machine features have relatively higher weights than the other machine features, as they are sparser.

Generating X^h We construct $p(x^h|x^m)$ so the machine features tell us which human features may be positive without giving us too much information about x_h . For each dimension of x_h , we set the weights of the corresponding logistic regression model to predict $p(x_d^h|x^m)$ so that exactly 1 dimension, d' randomly sampled from $\{1, \dots, D_m\}$, has weight $4.5 + \epsilon$, and all other dimensions have weight $\epsilon \sim N(0, 0.01)$. We set the bias term $w_d^0 = -4.5 + \epsilon$. This has the effect that if $x_{d'}^m = 1$, $p(x_d^h = 1)$ is approximately 0.5, and otherwise it is low. We construct the probability distribution for each dimension of X_d^h in this way, sampling d' without replacement. To generate the dataset, we sample X^h from this distribution, after first sampling X^m as described below.

Generating X^m We define a sampling procedure for X^m so that each instance has a single human feature that may be positive (and those should be different for different instances). We achieve this by setting exactly 1 of the sampled d' used in the previous step to be set to 1 for each instance. For the remaining features not in the set of sampled d' , we sample them with $p = 0.5$. Since each human feature has a distinct d' that signals it may be positive, sampling exactly one of these guarantees that the probability of the human features being on will only be high (around 50%) for 1 human feature.

Generating Y Finally, to generate Y from the sampled X , we generate the weights of a logistic regression model as follows, then sample Y from the probability distribution $p(Y|X)$ defined by this model. We do this so the human features are important when they are positive. To achieve this, we set each weight for the human features to either $3 + \epsilon'$, or $-3 + \epsilon'$ with equal probability (this makes it so sometimes the human features make the label more likely to be positive, and sometimes more likely to be negative) where $\epsilon' \sim N(0, 0.1)$. We also set the weights of the machine feature indices selected in step 1 as d' in the same way. For the remaining dimensions of X^m , we sample weights at either $1 + \epsilon$ or $-1 + \epsilon$ so they still influence the prediction, but to a lesser degree. This is to balance the fact that each instance is likely to have multiple of these dimensions that are positive vs. a single one of the other 2 sets of dimensions, and we don't want any one set to dominate the prediction. We set the bias to 0. We repeat this procedure until the labels are approximately class balanced, with mean between 0.4 and 0.6.

1.2 EXPERIMENT

Hyperparameters We split the data, class-balancing labels, into train/validation/test set of sizes $\frac{2}{3}, \frac{1}{6}, \frac{1}{6}$ of the instances respectively. We set the hyperparameters to penalty: none, class weighting: none, and $B = 5$. Aside from these, we use identical hyperparameter settings to those described in the main paper.

Results: Our approach reaches near-optimal performance quickly, while standard feature selection improves much more slowly. Figure ?? shows the results, averaged over 10 random restarts, of our method compared to feature selection, the full set of features, and the machine features with a budget of 5 queries. The error bars denote standard deviations. *all-features* performs best with f1-score of 0.88, while *machine-only* performs worst with f1-score around 0.815. *entropy-selection* performs only slightly worse than *all-features* from the first query, with an f1-score around 0.87. This makes sense, as the dataset was generated so that exactly 1 human feature is both uncertain and impactful for each instance, and as such, is important to query. *feature-selection* has a worse starting f1-score, around 0.83, and is unable to match the performance of *all-features* within the 5 queries. It only achieves f1-score of around 0.86 by the end. This suggests that our proposed method works to identify features that are both impactful for predictive performance, and can also be only be imperfectly predicted from the machine features.

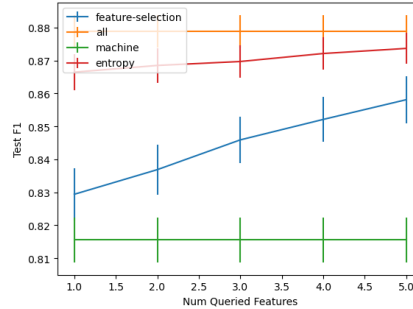


Figure 1: Test f1-score as a function of B for our proposed method, as well as baselines and upper bound. As expected, *all-features* performs best with f1-score around 0.88 and *machine-only* performs worst with f1-score around 0.815. *entropy-selection* performs almost as well as the upper bound from the first query with f1-score around 0.87, but *feature-selection* only begins to approach this performance after all 5 queries, with f1-score around 0.86.