# Feature Selection for Rare-Weak Features Using ANOVA Higher Criticism Thresholding
# Final Project Report

Roei Golan, 034812529

## 1 Introduction

Training data scarcity and quality have long been fundamental challenges in inference and learning. In modern machine learning and data analysis scenarios, a new challenge of feature rarity (aka sparsity) often arises: the number of available features is potentially vast, although only a few features may be useful while the rest are pure noise. Indeed, with the utilization of high throughput measuring devices for data collection, or the use of transfer learning practices resulting from the usefulness of foundation models ([1]), it is becoming common for practitioners to encounter feature vectors of dimensions up to 10's of thousands. On the other hand, the limited availability of training data is inherent to some inference tasks such as detecting rare diseases in medicine and rare market states in finance.

In the presented project, we build upon a method developed to handle feature sparsity from a statistical decision theory perspective, especially for cases where training data is scarce and noisy, cases for which the authors of the original method coined the term *rare/weak (RW) feature model* ([9]. We extend the method applicability from the binary classification to the multi-class setting, and evaluate its performance on modern datasets and use-cases.

This report is arranged in two parts. We begin by describing the RW feature model in the multi-class setting (2), and provide a description of the extended method (3). The second part describes all the experiments performed and analysis of the empirical results, first using synthetic data (4), followed by experiments made on real-world data (5).

## 2 Rare-Weak feature model in a multi-class setting

This section describes our feature (data) model for a rare/weak, multi-class classification problem. We begin by describing the binary problem analyzed in [9], and expand the model to a data model for the multi-class case.

In the binary case, we're given a $P$-dimensional dataset $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^P$ and $y_i \in \{0, 1\}$. We assume the data of each class is normally distributed around the class mean $\mu_c$ with some covariance matrix $\Sigma$ which is the same for both classes.

$$X_i \sim \begin{cases} \mathcal{N}(\mu_0, \Sigma), & y_i = 0 \\ \mathcal{N}(\mu_1, \Sigma), & y_i = 1 \end{cases}$$

For model simplicity, we assume $\Sigma = aI$, with $a$ some constant. I.e., the features are independent and cluster is isotropic around the mean.

denote $\mu_d \in \mathbb{R}^P$ the feature contrast vector $\mu_d = \mu_1 - \mu_0$. in the rare/weak (RW) feature model, the following holds:

- **Useful features are rare**. we suppose the feature contrast vector $\mu_d$ to be non-zero in only $D$ out of the $P$ features, where $\epsilon = \frac{D}{P}$ is small. I.e., the two class centroids coincide in most dimensions, and different only on $D$ out of $P$ features, which are obviously the useful features.
- **Useful features are weak (noisy)**. The non-zero elements of $\mu_d$ have the same amplitude $\mu$, which is assumed to be small compared to the variance of class points distribution around the class mean.

- **Training Data is scarce**. Another source of noise is the training set size, since the estimation error of the true class centroid is inversely proportional to the training set size. In the RW feature model, $n << P$

Throughout this project, we fixed the training set size to a small value (typically in the 20-50 samples per class range), and so we refer to $\epsilon$ and $\mu$ as the sparsity and strength parameters for the RW model.

Generalizing to the multi-class case, we simply increase the number of normal isotropic clusters from 2 to $K$. The clusters centroids are on the vertices of a $D$ dimensional hypercube in a $P$ dimensional space. The hypercube sides length is $\mu$.

Formally, we construct the class means as follows. given $K$ classes and $D$ useful features, let $\mu_i^{(k)}$ be the $i^{th}$ coordinate of the $k^{th}$ class mean $\mu^{(k)}$, and define $S$ to be the set of useful features. then:

$$\mu_i^{(k)} = \begin{cases} \mu \times \mathcal{U}\{-\frac{1}{2}, \frac{1}{2}\}, & i \in S \\ 0, & i \notin S \end{cases}$$

and we enforce the condition that each class centroid vector is unique - i.e., no two (or more) class centroids can occupy the same $P$-dimensional vector.

# 3 Method Description

Our feature selection method, which we refer to here as **HC-ANOVA**, can be seen as one natural extension of the binary case proposed in [9] to a multi-class setting, and is based on the **Diversity Pursuit (DP)** approach. In DP, we assume that a subset of the features (very small subset, per the RW feature model) is relevant for all classes, and the distribution of these features over the classes is diverse.

The HC-ANOVA is a supervised filter method, model free, and is based on well-understood ANOVA ([27], [19]) for feature importance and the Higher Criticism for detection and feature selection [8]. The first step is individually scoring the usefulness of every feature using standard ANOVA, followed by a second step of selecting a set of important features using the Higher Criticism threshold (HCT) [9].

In order to evaluate the quality of feature selection across datasets where the true identity of useful features is unknown, and compare it with other methods (mostly embedded) which are not model free, we pipeline the HC-ANOVA feature selector with a classifier and use the classification error as a proxy for the goodness of features. Following the ideology of [9, 15], even a simple classifier is competitive when fed with a good set of features. In this project, we use a simple nearest class centroid classifier, i.e., we associate to the sample the class whose mean vector minimizes the Euclidean norm or cosine similarity over the selected features. Other post-selection classification methods are possible, although empirically we didn't find any significant improvement over the centroid similarity (CS).

Formally, given a supervised classification task with $k$ classes, we have training data $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^P$ and $y_i \in \{1, \ldots, k\}$. Let $n_c$ be the number of training samples in class $c$, for $c = 1, \ldots, k$. As explained earlier, typically $n$ is very small while $k$ is larger than 2 (in the case $k = 2$ our method essentially reduces to [9]). The below steps are then applied:

Step I. **Per-feature ANOVA**. For every feature $j = 1, \ldots, P$,we compute a one-way ANOVA P-value in one of two methods:

– Parametric ANOVA: compute the per-feature statistic $F^{(j)}$:

$$SS^{(j)}_{within} := \sum_{c=1}^{k} \sum_{i\,:\,y_i=c} \left(x_i^{(j)} - \hat{\mu}_c^{(j)}\right)^2, \qquad \hat{\mu}_c^{(j)} = \frac{1}{n_c} \sum_{i\,:\,y_i=c} x_i^{(j)}$$

$$SS^{(j)}_{between} := \sum_{c=1}^{k} n_c \left(\hat{\mu}_c^{(j)} - \hat{\mu}^{(j)}\right)^2, \qquad \hat{\mu}^{(j)} = \frac{1}{n} \sum_{c=1}^{k} n_c \hat{\mu}_c^{(j)}$$

$$F^{(j)} := \frac{\frac{1}{k-1} SS^{(j)}_{between}}{\frac{1}{n-k} SS^{(j)}_{\text{within}}}$$

$F^{(j)}$ is the per-feature statistic and under the null (there is no difference in class means) is distributed with $F$ distribution with $k-1$ over $n-k$ degrees of freedom denoted $F_{k-1,n-k}$. The P-value $\pi^{(j)}$ is then given by:

$$\pi^{(j)} := \Pr\left[F_{k-1,n-k} \geq F^{(j)}\right].$$

– Non-Parametric ANOVA using the Kruskal-Wallis ([19]) test. compute the per-feature statistic $H^{(j)}$:

$$H^{(j)} := (n-1) \frac{\sum_{c=1}^{k} n_c \left(\bar{r}_c^{(j)} - \bar{r}^{(j)}\right)^2}{\sum_{c=1}^{k} \sum_{i\,:\,y_i=c} \left(r_i^{(j)} - \bar{r}^{(j)}\right)^2}$$

where the average within-class rank $\bar{r}_c^{(j)}$ and the average rank in the data $\bar{r}^{(j)}$ are given by:

$$\bar{r}_c^{(j)} = \frac{1}{n_c} \sum_{i\,:\,y_i=c} r_i^{(j)} \qquad \bar{r}^{(j)} = \frac{1}{2}(n+1)$$

Under the null hypothesis that samples median is the same for all classes, the per feature Kruskal-Wallis statistic is distributed with $\chi^2$ distribution with $k-1$ degrees of freedom:

$$H^{(j)} \sim \chi^2_{k-1}$$

The P-value $\pi^{(j)}$ is then given by:

$$\pi^{(j)} := \Pr\left[\chi^2_{k-1} \geq H^{(j)}\right].$$

Step II. **HC thresholding**. Denote by $\pi_{(1)}, \ldots, \pi_{(p)}$ the order statistics of the P-values $\{\pi^{(j)}\}_{j=1}^{P}$. The Higher Criticism of these P-values is given by [7]

$$HC^\star := \max_{1 \leq i \leq \frac{n}{2}, \pi_{(i)} > \frac{1}{n}} HC_i, \qquad HC_i := \sqrt{n} \frac{\frac{i}{n} - \pi_{(i)}}{\sqrt{\pi_{(i)}\left(1 - \pi_{(i)}\right)}} \tag{1}$$

Let $i^\star$ denote the index maximizing $HC_i$ in (1). The HC thresholding procedure selects the set of indices $\Delta^\star \subset \{1, \ldots, P\}$ defined as

$$\Delta^\star = \left\{j \,:\, \pi^{(j)} \leq \pi_{(i^*)}\right\}.$$

I.e., we select all the features with p-value smaller than the threshold p-value $\pi_{(i^*)}$.

Step III. **Classification**. Given a sample $x \in \mathbb{R}^P$, we classify it to the class whose mean is closest to $x$ when only selected features are considered, using cosine similarity as a measure. Namely, we classify $x$ to $\hat{c}(x)$ given by:

$$\hat{c}(x) := \underset{c\in\{1,\ldots,k\}}{\operatorname{argmin}} \frac{\left|\sum_{j\in\Delta^*} x^{(j)}\hat{\mu}_c^{(j)}\right|}{\sqrt{\sum_{j\in\Delta^*}(x^{(j)})^2}\sqrt{\sum_{j\in\Delta^*}(\hat{\mu}_c^{(j)})^2}}.$$

**Alternative extension of binary HC** Another (more trivial) possible extension of [9], is the one-vs-all (OvA) approach, which assumes that each class has a characteristic set of features that discriminate it from the rest. Intuitively, the OvA approach is less plausible when considered for a rare-weak data, since the number of useful features should at least scale with the number of classes. The OvA approach is utilized by treating the $k$ classes problem as $k$ binary problems, use [9] method to identify features for each binary problem, and then collect all the features to perform a multi-class classification. We implemented an HC-OvA feature selector and coupled with a simple CS classifier, we included it in our trials, but found it to be inferior to the HC-ANOVA (DP approach) in most cases.

# 4 Empirical Results - Synthetic data

This section evaluates our method and benchmark it on synthetically generated datasets, which allow precise control of data parameters such as sparsity, signal strength, data dimension and number of classes. The first part includes experiments on synthetic data corresponding to our RW features model (section 2) - namely linearly separable data, where each class data is a single isotropic cluster. The goal is to test the method across various data parameters. An inherent part of the RW data model is the small amount of training data, and in all experiments the train set size is typically 30-50 samples per class.

The second part deals with non-linear data, and its goal is to demonstrate the ability to use any set of p-values which serve as feature scores in the HC thresholding process - a functionality supported in the code package developed in this project.

## 4.1 Normally distributed clusters

**Exploring performance over the sparsity-strength space** In [6], Donoho & Jin provide a theoretical analysis of the rare-weak (RW) model, deriving a closed form expressions for the region of the sparsity-strength space where classification based on ideal HCT will be successful (and show it is identical to the area where classification based on the ideal threshold on features p-values is successful). However, their analysis applies to the binary classification problem and a Z-test based HCT, while this project explores a more general use of HCT. A similar analysis of the general case is beyond the scope of this project, and in order to understand the regions of sparsity-strength space for which HCT feature selection is successful or not (and competitive with other FS methods) we empirically scanned the space, using a grid of sparsity-strength, as follows:

- For sparsity, we adopted the terminology of Donoho & Jin ([6]) and control it using $\beta$, where the fraction of useful features $\epsilon$ out of all features $p$ is given by $\epsilon = p^{-\beta}$. In our grid, $\beta \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$. As $\beta$ grows, $\epsilon$ gets smaller and the problem gets more sparse and difficult.
- Strength is controlled by the inter-centroid distance $\mu$ along a useful feature axis and its ratio to the noise - the variance of class points around the centroid. We set the variance to 1 and select $\mu$ to cover the range from very difficult problem ($\mu_{min}$, where tested classifiers performed at chance level) to very easy problem ($\mu_{max}$, where most tested classifiers achieved perfect accuracy). We then divided the range $[\mu_{min}, \mu_{max}]$ to 100 points that define the strength grid.

For each point on this 600 points 2D grid we generated a multi-class classification problem per our data model as described in section 2. We tested our HC-ANOVA method (DP) against a plethora of classic and modern classifiers on this problem: Naive Bayes, SVM, logistic regression, KNN, decision tree and two ensemble methods: Random Forest and Gradient Boosting classifier. Of all these, L1 regulated linear classifiers (logistic regression and linear SVM) performed best and are reported here. Table 1 summarizes all the classifiers reported in this section (4.1).

| **Classifier** | **Description** |
|---|---|
| CS all features | The most simple classifier based on cosine similarity to classes centroids. Each class centroid is set as the mean of class training samples, using all available features. |
| CS + HC-ANOVA | Our Diversity Pursuit method as described in section 3. Compute class centroid and cosine similarity using only features selected by HCT on ANOVA P-values. |
| CS + HC-OvA (t-test) | This is a naive extension of the use features selected by HC thresholding on P-values for difference of means (t-tests) between each class and the rest of classes. |
| SVM + Lasso | Linear SVM with L1 regularization. |
| LR + Lasso | logistic regression with L1 regularization. |

Table 1: A summary of classifiers benchmarked on section 4.1

The experiment parameters (for each grid point) are described in table 2 below:

| | |
|---|---|
| number of experiments in each grid point | 30 |
| number of train samples per class | 50 |
| number of test samples per class | 500 |
| number of classes | 5 |
| feature dimension | 5000 |
| Linear classifiers tuning | Grid search using 3-fold CV. |
| Linear classifiers regularization grid | {1e-3,1e-2,1e-1,1,1e1,1e2} |

Table 2: Exploring the sparseness-strength grid: experiment parameters.

We calculated mean and standard deviation of the classification accuracy for each of the classifiers, and plot the accuracy curves in figure 1. As expected, the problem gets harder as there are less useful features ($\epsilon$ gets smaller, plots (a) to (f)) and the all classifiers require stronger signal to achieve the same level of accuracy. There is no performance difference between the classifiers in the most dense case (plot (a)), but as the problem gets sparser the accuracy of the most simple classifier (CS all features) deteriorates fastest, indicating the necessity of noise filtering by feature selection in this problem. While both HCT methods show similar accuracy and dominates the Lasso regulated classifiers in lower levels of sparsity (plots (b),(c)), the picture is changing as the problem turns even more sparser. The HC-ANOVA (DP) method performs best of all when the signal is weak, and the L1 regulated linear classifiers outperform the HC-OvA classifier (plots (d) to (f)). This is because the OvA method, by nature, allows more features (many of them uninformative) to pass through the filter and thus introduces more noise into the classification.

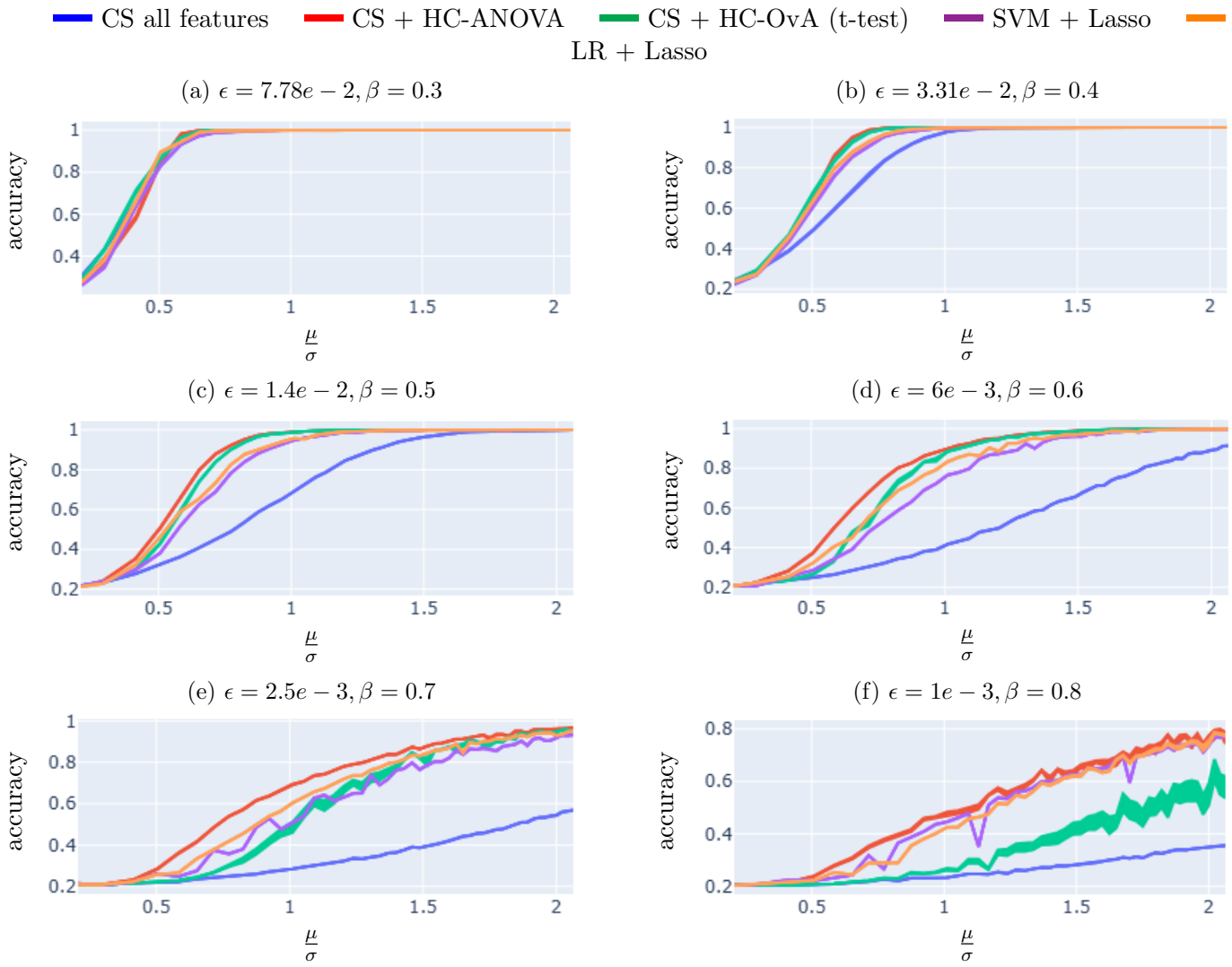


Figure 1: Evaluating accuracy on the sparsity-strength grid. Figures a-f represent different $\epsilon$ values (sparsity), while the inter-class distance $\mu$ (strength) span the horizontal axis in each figure. Curve thickness represents standard deviation of the mean accuracy.

**scanning the data dimension** Scanning the performance of the suggested method in the sparsity-strength space revealed that it excels in weak signals, over a wide range of sparsity levels ($\beta \geq 0.4$). We therefore focused on this region to test performance across a range of feature dimensions. We set $\beta = 0.6$, and scanned the feature dimension across 5 values of $P$:

$$P \in [100, 500, 1000, 5000, 10000]$$

Since keeping $\beta$ fixed and changing $P$ increases the sparsity ($\epsilon$ gets smaller), we initialized $\mu = 0.6$ and let it grow logarithmically with $P$ to compensate for that added sparsity. For each feature dimension we generated a synthetic classification problem as described in section 2. The train and test sizes, as well as the hyper-parameters of the tested classifiers are as described in table 2. We analyzed both 4 and 10 classes problems. We simulated each problem 40 times (each time generating a new, random data set) and measured the classification accuracy. The mean accuracy of each classifier is shown in table 3 with the best result in bold.

| $P$ | 4 classes | | | | | 10 classes | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | CS all features | CS + HC-ANOVA | CS + HC-OvA | SVM + Lasso | LR + Lasso | CS all features | CS + HC-ANOVA | CS + HC-OvA | SVM + Lasso | LR + Lasso |
| 100 | 0.36 ±0.003 | **0.41 ±0.008** | 0.36 ±0.003 | 0.36 ±0.005 | **0.42 ±0.006** | 0.19 ±0.002 | **0.25 ±0.003** | 0.20 ±0.002 | 0.18 ±0.002 | 0.21 ±0.003 |
| 500 | 0.39 ±0.003 | **0.57 ±0.005** | 0.41 ±0.008 | 0.43 ±0.007 | 0.51 ±0.006 | 0.21 ±0.002 | **0.42 ±0.002** | 0.29 ±0.007 | 0.24 ±0.002 | 0.31 ±0.003 |
| 1000 | 0.38 ±0.003 | **0.61 ±0.006** | 0.43 ±0.013 | 0.49 ±0.006 | 0.56 ±0.005 | 0.21 ±0.002 | **0.50 ±0.003** | 0.37 ±0.010 | 0.28 ±0.003 | 0.37 ±0.002 |
| 5000 | 0.40 ±0.004 | **0.80 ±0.005** | 0.70 ±0.022 | 0.65 ±0.007 | 0.73 ±0.005 | 0.23 ±0.002 | **0.77 ±0.002** | 0.70 ±0.009 | 0.48 ±0.004 | 0.60 ±0.003 |
| 10,000 | 0.40 ±0.003 | **0.89 ±0.003** | 0.83 ±0.017 | 0.75 ±0.005 | 0.81 ±0.004 | 0.23 ±0.001 | **0.87 ±0.002** | 0.84 ±0.003 | 0.59 ±0.004 | 0.70 ±0.003 |

Table 3: Classification accuracy for different feature dimensions

**Evaluating feature selection accuracy** As can be seen in table 3, the HC-ANOVA feature selection coupled with a simple CS classifier is better then L1 regulated classifiers for almost all $P$ values. This is of course a result of better feature selection, as the two other CS classifiers underperformed. It's insightful to see where exactly the HC-ANOVA method is better. using a synthetic dataset where the identity of useful features is known allows the assessment of the precision of the identification of features. We recorded the actual useful features identities in each classification experiment and evaluated the recall, precision, and F1 score of the feature selection for each classifier. The features selected by the HCT classifiers are these left after the thresholding process. For the Lasso regularized classifiers, the selected features are the ones for which the coefficient is not zero. The mean metrics (for all experiments in a given data dimension) are reported in table 4.

| $P$ | recall | | | | precision | | | | F1 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HC-ANOVA | HC-OvA | SVM L1 | Log-Reg L1 | HC-ANOVA | HC-OvA | SVM L1 | Log-Reg L1 | HC-ANOVA | HC-OvA | SVM L1 | Log-Reg L1 |
| 4 classes | | | | | | | | | | | | |
| 100 | 0.93 | **1** | 0.98 | 0.99 | **0.5** | 0.07 | 0.11 | 0.21 | **0.57** | 0.13 | 0.15 | 0.35 |
| 500 | 0.88 | **1** | 0.9 | 0.99 | **0.84** | 0.06 | 0.3 | 0.14 | **0.85** | 0.1 | 0.26 | 0.24 |
| 1000 | 0.83 | **0.99** | 0.87 | 0.98 | **0.86** | 0.1 | 0.33 | 0.13 | **0.84** | 0.14 | 0.29 | 0.23 |
| 5000 | 0.85 | 0.9 | 0.94 | **0.97** | **0.92** | 0.5 | 0.18 | 0.18 | **0.88** | 0.59 | 0.2 | 0.3 |
| 10000 | 0.87 | 0.88 | 0.93 | **0.96** | **0.94** | 0.68 | 0.2 | 0.23 | **0.9** | 0.74 | 0.21 | 0.37 |
| 10 classes | | | | | | | | | | | | |
| 100 | **1** | **1** | 0.97 | **1** | **0.72** | 0.1 | 0.13 | 0.09 | **0.83** | 0.19 | 0.18 | 0.17 |
| 500 | **1** | **1** | 0.98 | **1** | **0.86** | 0.13 | 0.28 | 0.05 | **0.93** | 0.22 | 0.3 | 0.09 |
| 1000 | 0.99 | **1** | **1** | **1** | **0.9** | 0.2 | 0.18 | 0.04 | **0.94** | 0.32 | 0.2 | 0.08 |
| 5000 | **1** | **1** | 0.98 | **1** | **0.93** | 0.44 | 0.15 | 0.05 | **0.97** | 0.59 | 0.17 | 0.1 |
| 10000 | 0.99 | 0.99 | **1** | **1** | **0.95** | 0.62 | 0.062 | 0.06 | **0.97** | 0.76 | 0.08 | 0.12 |

Table 4: Feature selection metrics

Table 4 shows that HC-ANOVA is more selective than other methods, resulting in degradation in recall (at least for small number of classes). However, its higher precision allows it to filter most of the noise, resulting in better accuracy. In rare-weak (RW) feature model, useful features are each too weak to be of much use on their own ([9]). It's reasonable to assume that a sufficient level of recall is required for successful classification, but given enough useful features identified, it's better to filter more aggressively and mitigate the noise. All classifiers in this experiment achieve high recall rate, but the accuracy is correlated with precision/F1, indicating this is the limiting factor.

### 4.2 Departing from linearly separable data

The method described in Section 3 applies to linearly separable data where each class points are arranged in a separate isotropic (or at least roughly isotropic) cluster, and is based on class means separation - identifying features where class means differ enough to allow separation using P-values which capture the difference in means (parametric / nonparametric ANOVA). However, higher criticism thresholding (HCT) can be applied to any statistical test that captures differences in the class-conditioned values distribution for each feature, and thus identify useful features. This is done by replacing the ANOVA p-values with the appropriate test p-values.

This section is somewhat of a detour from the main subject of the project, but we considered it worthwhile to demonstrate the integration of HCT with other statistical tests that may be useful for feature selection, with hope it will trigger interest and further research.

In order to demonstrate it, we experimented with a binary classification problem using synthetic dataset where in the useful features points from class 0 are below the median and class 1 points are above the median of a multi-dimensional isotropic Gaussian. As in the previous data, in the useless (noise) features each class points are distributed normally with constant variance around the origin. An example for a pair of useful and useless features is illustrated in figure 2.

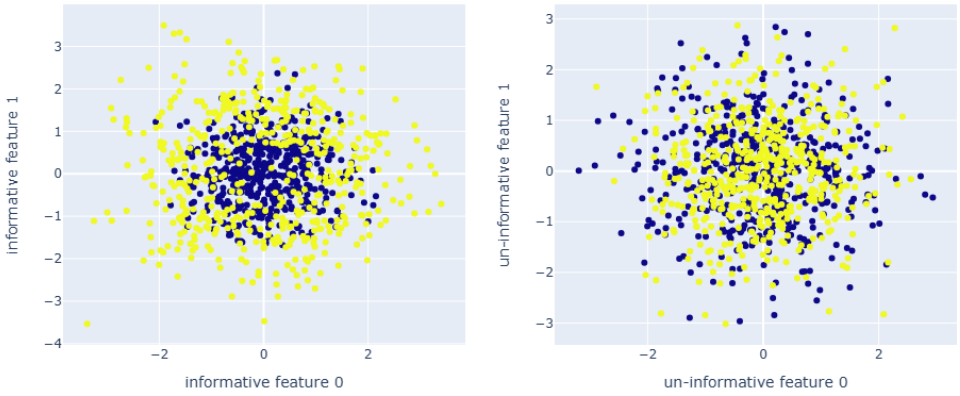


Figure 2: Nested concentric circles data example. Left: pair of informative features. Right: pair of un-informative features. Points of different classes have different colors

Clearly, this is not a linearly separable data (at least not without a non-linear transformation. e.g. to polar coordinates). Both class means coincide (in the origin in this case) and use of ANOVA based test is useless (and so the use in centroid similarity). Instead, we used the 2-sample Kolmogorov-Smirnov (KS) test ([11]), which compares the empirical distribution function of two samples, and is sensitive to both location and shape of the distributions. For each feature $j$ in the data, we created 2 samples:

$$S_0^j = \{X_i^j, \forall i : y_i = 0\}, S_1^j = \{X_i^j, \forall i : y_i = 1\}$$

The KS test is then applied on these two samples to obtain a p-value $\pi^{(j)}$, and the set of p-values corresponding to all features can be plugged in step 2 of the method as described in section 3.

We repeated the experiment described in 4.1, evaluating binary classification accuracy across multiple data dimensions. We tested non-linear classifiers (SVM with RBF kernel, random forest), both with and without HCT feature selection as a pre-training feature selection step. Classifiers hyper parameters were tuned using 3-fold CV grid search for each experiment, with parameters as described in table 5 below. The RF classifier itself is a popular embedded feature selection method ([2]) and as another competitor FS method we evaluated the popular uni-variate filter method of scoring each feature based on its mutual information (MI) with the target ([13], [23]) and then selecting the highest $K$ scores, where $K$ is chosen to maximize accuracy using cross validated grid search ($K$ grid also in table 5). We pipelined that method as a pre-training FS step with the SVM classifier and report its accuracy as well.

| | |
|---|---|
| number of train samples per class | 250 |
| number of test samples per class | 500 |
| SVM kernel grid | {RBF, polynomial} |
| SVM regularization grid | {1e-3,1e-2,1e-1,1,1e1,1e2} |
| RF number of trees grid | {10,20,50,100} |
| RF trees depth grid | {3,5,7,9} |
| MI scores $K$ grid | {5,10,20,50} |

Table 5: Experiment parameters for non-linear data classification.

| $P$ | SVM | Random Forest (RF) | SVM + Select K features with highest MI | SVM + HC on KS p-values | RF + HC on KS p-values |
|---|---|---|---|---|---|
| 100 | 0.585 ±0.009 | 0.805 ±0.006 | 0.833 ±0.005 | **0.842 ±0.009** | 0.820 ±0.005 |
| 500 | 0.523 ±0.005 | 0.825 ±0.007 | 0.819 ±0.011 | **0.879 ±0.006** | 0.863 ±0.004 |
| 1000 | 0.505 ±0.008 | 0.808 ±0.004 | **0.880 ±0.006** | 0.873 ±0.018 | 0.863 ±0.005 |
| 5000 | 0.492 ±0.003 | 0.813 ±0.005 | 0.874 ±0.004 | **0.908 ±0.003** | 0.880 ±0.003 |

Table 6: Classification accuracy in the binary, non-linear classification task, for different feature dimensions. Best metrics in each row in **bold**.

As can be seen in table 6, adding a HC based feature selector in the classification pipe always improved the accuracy of the classifier (i.e. SVM/RF without feature selection vs. SVM/RF with HCT feature selection), demonstrating the HC feature selector ability to identify the informative features. Comparing the accuracies of a pipeline with HCT feature selection to the "select K best MI score" shows comparable or slightly better accuracies in the HCT based pipeline, without the need to tune K using cross validation (and with a negligible computation cost, compared to the MI method).

As done in previous section we measured the feature selection metrics of each method to gain a deeper understanding of the classifiers' performance. The Random Forest features are identified using the feature importance (accumulated information gain in trees nodes for each feature) assigned to each feature during training. The metrics are summarized in table 7. Note that also reported is the mean number of features selected by each method for each data dimension, and the actual true number of useful features (number in parentheses in the $P$ column).

We believe the important insight from this analysis is the vulnerability of the "select $K$ best" methods to the choice of $K$, compared to adaptivity of the HCT method. In this experiment, a "not good enough" value of $K$ chosen either degraded the recall of the MI feature selector (for $P = 500, 1000$) or the precision (for $P = 100, 5000$). Indeed, both HCT and "select $K$ best" are thresholding methods, but "select $K$

best" is trying to estimate the threshold using cross-validation, and is thus susceptible to the grid choice.

| | number of features used | | | recall | | | precision | | | F1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P$ | HC | MI | RF | HC | MI | RF | HC | MI | RF | HC | MI | RF |
| 100 (6) | 6.9 | 10 | 100 | **1** | **1** | **1** | **0.92** | 0.6 | 0.06 | **0.95** | 0.75 | 0.11 |
| 500 (12) | 13.1 | 9.5 | 466 | **1** | 0.47 | **1** | **0.92** | 0.92 | 0.03 | **0.96** | 0.57 | 0.05 |
| 1000 (16) | 21.4 | 10 | 603.9 | 0.98 | 0.71 | **1** | 0.84 | **1** | 0.03 | **0.88** | 0.83 | 0.05 |
| 5000 (30) | 31.3 | 50 | 1235.9 | 0.95 | 0.98 | **1** | **0.92** | 0.59 | 0.03 | **0.93** | 0.73 | 0.06 |

Table 7: Non-linear classification task feature selection metrics. The number in parentheses for each value of $P$ is the number of useful features in the data.

# 5 Real World Data

This section includes our method evaluation on real world datasets. We explore 3 different use cases: classification of tabular data (gene expression micro-array data) and two domain adaptation/transfer learning cases: text classification using pre-trained LLM embeddings, and image classification using embeddings from pre-trained vision models. Gene expression classification tasks may benefit from our method since they are often very high dimensional and scarce in data ([24]). Classification using text/image embeddings is also a target use case from the following reasons. First, embeddings from modern large language models or vision models are high dimensional, with typical size of 1000 to 1000's of features ([30], [22]). Second, large vision models are trained on a very large number of classes and generalize to a diversity of visual objects ([32]), and LLMs are trained on vast and diverse corpora of text, intended to generalize effectively across a wide range of NLP tasks ([3]). When the downstream classification task is very fine-grained, most features in the embeddings might be irrelevant and useful features will be rare. Lastly, classes in the embedding space are typically linearly separable, with similar data clustered together ([17], [14]), and so ANOVA p-values for features scoring is suitable.

## 5.1 Gene Expression Data

Gene expression datasets typically comprise gene expression levels across different experimental conditions or biological samples. Classification these datasets is the task of classifying genes or samples into distinct categories based on their expression patterns. This classification plays a pivotal role in tasks such as disease diagnosis, drug discovery, and understanding gene regulatory networks. Challenges in gene expression classification involves issues as small samples, feature selection, class imbalance, and interpretability of models.

In order to test our method on multi-class, gene expression data sets, we've selected a few sets from the CuMiDa ([10], `https://sbcb.inf.ufrgs.br/cumida`) database of curated gene expression classification datasets. Our criteria for choosing the sets were two fold: first, we looked only at multi-class classification problems. Second, the authors (and curators) of CuMiDa reported baseline classification accuracy for each dataset. We assume easy problems don't suffer from the rare-weak features problem, and so chosen the datasets with the lowest baseline accuracy.

For each dataset, we performed 30 classification experiments, each one with a different split of the data to train and test sets. The train/test ratio was fixed to 80%/20% with a minimum of 3 samples per class for classes with very few samples (see the samples per class distribution in table 9). We used the centroid

similarity (CS) classifier using all the features as a baseline method and then used it in conjunction with HC based FS methods (CS+HC-ANOVA, CS+HC-OvA - as was done in the synthetic data experiments in section 4) and compared performance to a range of classifiers. In each experiment, the SVM/logistic regression classifiers were tuned on the train set using a 3-fold CV grid search with parameters as reported on table 8.

| | |
|---|---|
| number of experiments in each dataset | 30 |
| SVM kernel grid | {RBF, polynomial} |
| Linear SVM/Logistic regression regularization grid | {1e-3,1e-2,1e-1,1,1e1,1e2} |
| accuracy metric | balanced accuracy |

Table 8: Experiments parameters for gene expression classification

The results are summarized in table 9. Since most datasets are highly unbalanced, we used balanced accuracy as the performance metric. For each dataset the top row shows mean (across all splits) test set balanced accuracy for each classifier, and bottom row shows mean percentage of features used by each classifier, out of the total number of features (in parenthesis). The results show better performance of the suggested method (HC-ANOVA) in three out of four datasets. Judging by the number of features discovered by the HC-ANOVA, the useful features are indeed rare, with at most 1.25% of the features discovered by the best performing classifier. Interestingly, in each dataset the best performing classifier is the most selective one (feature wise). This suggests that the useless features are noisy, hence learning and inference on them degrades the accuracy.

| Dataset | Num. Classes (samples per class) | CS All features | CS HC-ANOVA | CS HC-OvA | SVM (Lasso) | SVM | LR (Lasso) |
|---|---|---|---|---|---|---|---|
| Breast_GSE26304 | 3 | 0.39 | **0.59** | 0.48 | 0.47 | 0.34 | 0.48 |
| (33637) | (42,36,31) | (100%) | **(0.021%)** | (1.561%) | (0.324%) | (100%) | (0.119%) |
| Breast_GSE10797 | 3 | 0.69 | **0.83** | 0.78 | 0.78 | 0.78 | 0.81 |
| (22277) | (28,28,10) | (100%) | **(1.14%)** | (3.762%) | (1.522%) | (100%) | (4.076%) |
| Ovary_GSE6008 | 4 | 0.77 | **0.80** | 0.76 | 0.74 | 0.75 | 0.77 |
| (22283) | (41,37,12,8) | (100%) | **(1.252%)** | (4.209%) | (22.869%) | (100%) | (18.184%) |
| Leukemia_GSE71449 | 3 | 0.68 | 0.91 | 0.76 | **0.95** | 0.78 | 0.84 |
| (52200) | (20,18,6) | (100%) | (0.533%) | (29.284%) | **(0.155%)** | (100%) | (5.084%) |

Table 9: Gene expression classification results. Top row - test set balanced accuracy (best in bold). Bottom row - percentage of features used by the classifier (least in bold).

### 5.2 Text classification with text embeddings

Large Language Models (LLMs) are recognized as powerful tools for generating text embeddings, which are dense vector representations capturing semantic relationships between words, sentences, or documents. These embeddings are widely used in various NLP tasks, such as text classification, retrieval-augmented generation (RAG), and semantic search. By encoding text into high-dimensional vectors, LLMs enable downstream models to leverage pre-trained knowledge, often achieving SoTA performance with minimal task-specific data ([25]). For instance, embeddings produced by models like GPT excel in fine-grained classification tasks, while also serving as effective inputs for retrieval systems in RAG workflows ([20]). This capability underscores their significance in modern NLP pipelines.

### 5.2.1 Hierarchical patents classification

In this experiment we tested our method on the task of patent classification. The main reason for selecting this task is that the hierarchical nature of patent labeling enables us to control how fine grained the classification is, and study our method behavior on different granularity levels, while the text input remains the same. We used a proprietary dataset that contains 200,000 abstracts of US patents, and for each patent we have a unique ID, the abstract itself and the CPC codes, from which we derive the labels for classification. Below is a short description of CPC codes, which is required to understand the different classification hierarchies, and two examples for patents descriptions and their codes (table 10).

In the **Cooperative Patent Classification (CPC)** system, alphanumeric codes are used to label patent documents based on their technical content. These codes serve a hierarchical classification purpose, beginning with a coarse level and refining into more specific categories. The first part of the CPC code, which includes the **section**, **class**, and **sub-class** (e.g., A61K), is used for coarse classification, identifying broad technical fields such as medical or chemical inventions. Within this section-class-subclass combination, **groups** are used as labels for a finer level of classification, narrowing down to more specific technologies or processes. For even greater precision, **sub-groups** are employed to classify inventions at the finest level, providing detailed categorization based on very specific aspects of the invention. This multi-tiered approach allows for a structured and granular method of patent classification, facilitating efficient patent searching and analysis.

| **Patent Abstract** | **Codes** |
|---|---|
| Various embodiments for deriving a motion vector predictor for a current block are provided. At least one first candidate of a predictor of the motion vector for the current block is derived from a first motion vector of a first block that is adjacent to the current block in the current picture. At least one second candidate of the predictor is derived from a second motion vector of a second block that is included in a picture that is different from the current picture when a number of the first candidate is less than a maximum number... | H04N 19/105<br>H04N 19/139<br>H04N 19/172<br>H04N 19/176<br>H04N 19/46<br>H04N 19/513<br>H04N 19/52<br>H04N 19/70 |
| A battery array frame according to an exemplary aspect of the present disclosure includes, among other things, a frame body, and a thermal fin including a body embedded in the frame body and a leg that extends outside of the frame body. The thermal fin is flexible between a first position in which the leg is spaced farther from a surface of the frame body and a second position in which the leg is spaced closer to the surface of the frame body. | B60L 50/64<br>H01M 2/1077<br>H01M 10/0525<br>H01M 10/613<br>H01M 50/20<br>H01M 50/211<br>Y02E 60/10<br>Y02E 60/50<br>Y02T 10/70 |

Table 10: Examples for patents abstracts and their corresponding codes.

**Experiment description** We performed 3 classification experiments, covering the most fine-grained 3 levels in the CPC hierarchy: sub-class, group and sub-group. For each level, we sampled 50 sets of 3 labels corresponding to that level. Each set defines a separate classification task. For each set, we extracted from the dataset all the patents corresponding to the labels. As can be seen in table 10, most patents have many CPC codes (multi-labeled) and so a patent can belong to more than one class in the classification task, and we removed these patents from the data. The data size in each task was 300 patents per class, and the train\test split is 30\270 patents per class. We embedded the patents using 4 embedding models, with the main motivation to explore different embedding dimensions. The embeddings models are summarized in table 11:

| Model Name | Commercial/open source | Embedding Dimension |
|---|---|---|
| text-embedding-3-large | Commercial - OpenAI | 3072 |
| text-embedding-3-small | Commercial - OpenAI | 1536 |
| embed-english-v3.0 | Commercial - Cohere | 4096 |
| GPT-2 | open source ([25]) | 768 |

Table 11: Embedding models used for patent classification tasks

**Embeddings classification results** The results are summarized in table 12 below. As expected, when the classification becomes more fine-grained (from "sub-class" to "group" to "sub-group") the accuracy drops, across all embeddings and all classifiers. The percentage of features selected by the HCT based classifiers and the Lasso regularized logistic regression drops as well, indicating greater class similarity and less useful features. Apart from the GPT-2 embeddings, the accuracies across classifiers are very close. It's interesting, though, that with the GPT-2 embeddings our method noticeably underperformed in the coarse classification tasks (sub-class, group), but nearly closed the gap in the most fine-grained task, when the number of useful features is indeed the smallest.

| Level | sub-class | | | | group | | | | sub-group | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| classifier | CS + HC-ANOVA | CS + HC-OvA | SVM | Log-Reg-Lasso | CS + HC-ANOVA | CS + HC-OvA | SVM | Log-Reg-Lasso | CS + HC-ANOVA | CS + HC-OvA | SVM | Log-Reg-Lasso |
| Cohere (4096) | 0.826 (28.6%) | 0.83 (35.3%) | **0.835** (100%) | 0.816 (9.52%) | 0.77 (21.01%) | 0.77 (28.12%) | **0.772** (100%) | 0.75 (8.79%) | 0.64 (13.94%) | **0.65** (24.95%) | 0.644 (100%) | 0.621 (9.45%) |
| OpenAI-Large (3072) | 0.85 (14.97%) | 0.85 (19.21%) | **0.86** (100%) | 0.84 (12.73%) | 0.77 (12.37%) | 0.78 (16.8%) | **0.79** (100%) | 0.78 (11.1%) | 0.65 (6.18%) | **0.66** (12.01%) | **0.66** (100%) | 0.63 (10.03%) |
| OpenAI-Small (1536) | 0.83 (26.11%) | 0.83 (35.61%) | **0.84** (100%) | 0.81 (17.51%) | 0.746 (18.1%) | 0.754 (26.5%) | **0.756** (100%) | 0.73 (13.8%) | 0.64 (14.39%) | **0.65** (30.66%) | **0.65** (100%) | 0.62 (14.65%) |
| GPT-2 (768) | 0.7 (39.19%) | 0.68 (47.79%) | **0.78** (100%) | 0.76 (24.35%) | 0.64 (29.04%) | 0.63 (37.5%) | **0.72** (100%) | 0.7 (26.3%) | 0.55 (20.05%) | 0.53 (42.84%) | **0.58** (100%) | 0.55 (19.14%) |

Table 12: Patents abstracts embedding classification results. For each embedding type and classification level, the top number shows the mean accuracy across all experiments in that level, and the bottom number (in parentheses) shows the percentage of features selected by each classifier. The number under the embedding model name is the embedding dimension.

**class means distances across different levels** As a demonstration and reinforcement to our assumption that the features will be weaker and sparser as the task requires classification between finer code levels, we performed the following analysis. Using our classification tasks data as described above, embedded using OpenAI's text-embedding-3-large model, we computed for each task data the 3 class centroids (mean of class samples) and then computed the 3 euclidean distances within this set of centroids. For each classification level (coarse to fine) we collected these distances (from all 50 classification tasks) to form a distribution of the inter-class distances within this level. The histograms of these distributions are presented in figure 3. As can be seen, as the classification is more fine-grained, the distribution of inter-class distances shifts left, meaning that class centroids are closer (which may result from the number of informative features getting smaller, class difference along a useful dimension getting smaller, or both).

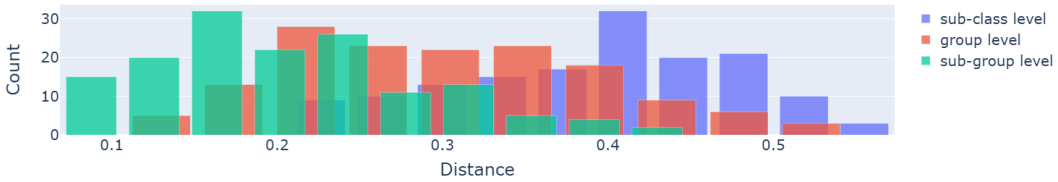

Figure 3: Distribution of class means distances across CPC codes levels

### 5.3 Image Classification with pre-trained models

A common practice in image classification (especially with limited data) is to use a pre-trained large vision model as a feature extractor (embedding model), and then train a simple classifier on the embeddings ([26], [32]). In this experiment we benchmarked our method against other popular classifiers in this use case.

When considering datasets for benchmarking our method, we focused on Fine Grained Visual Classification (FGVC) sets, which typically comprise classes of similar objects. We focused on 4 well known FGVC sets: Oxford-flowers ([21]), Stanford-dogs ([16]), Stanford-cars ([18]) and CUB-200 (Caltech-UCSD Birds, [31]). **A major challenge was to identify RW classification problems within these sets**. We tried to identify such problems in two ways: empirically, and by domain knowledge.

Empirically, we iteratively sampled a large group of classes from a dataset, and performed a multi-class classification task on this group, using a pre-trained model as feature extractor and L1-regularized logistic regressor as a classifier. From the test results confusion matrix, we identified the subsets of classes with larger classification errors. Assuming it indicates high class similarity and a potential RW problem, we took these subsets for benchmarking our method.

By applying domain knowledge, we reviewed the classes in each dataset, trying to identify groups of similar classes (e.g. same car model from different years in Stanford-cars, few species from the same family of birds in CUB-200 etc.). We verified visually which groups are hard to classify, and considered these groups as potential RW problems as well.

Surprisingly, almost all subsets we found were not RW, judging by the number of features discovered during the benchmark (both by our method and by the L1 regularized classifiers we tried). The only mildly RW subsets we found all belong to the CUB-200 ([31]) dataset, and below we detail the benchmark process and results. In all non-RW problems we tried, the accuracies (of all classifiers) were generally higher than in the RW problems, and our method underperformed or at best was equivalent to the other classifiers tested.

**Classification of bird species from CUB-200** The Caltech-UCSD Birds 200 (CUB-200, [31]) dataset is a popular benchmark for fine-grained image classification, containing 200 bird species (classes), with 30 images (train+test) per class. Most bird species can be further grouped into bird families, with few to dozens of species per family. All the RW problems we found and report are from different families, and all involve classification of bird species that belong to the same family. Figure 4 shows image samples from 3 of the problems. Each row represents a family (a separate 3-class classification task), and the bird species (2 examples per class) comprising this task. At least from human error perspective, it's clear that telling families from one another is relatively easy, while classifying species within a family is challenging.

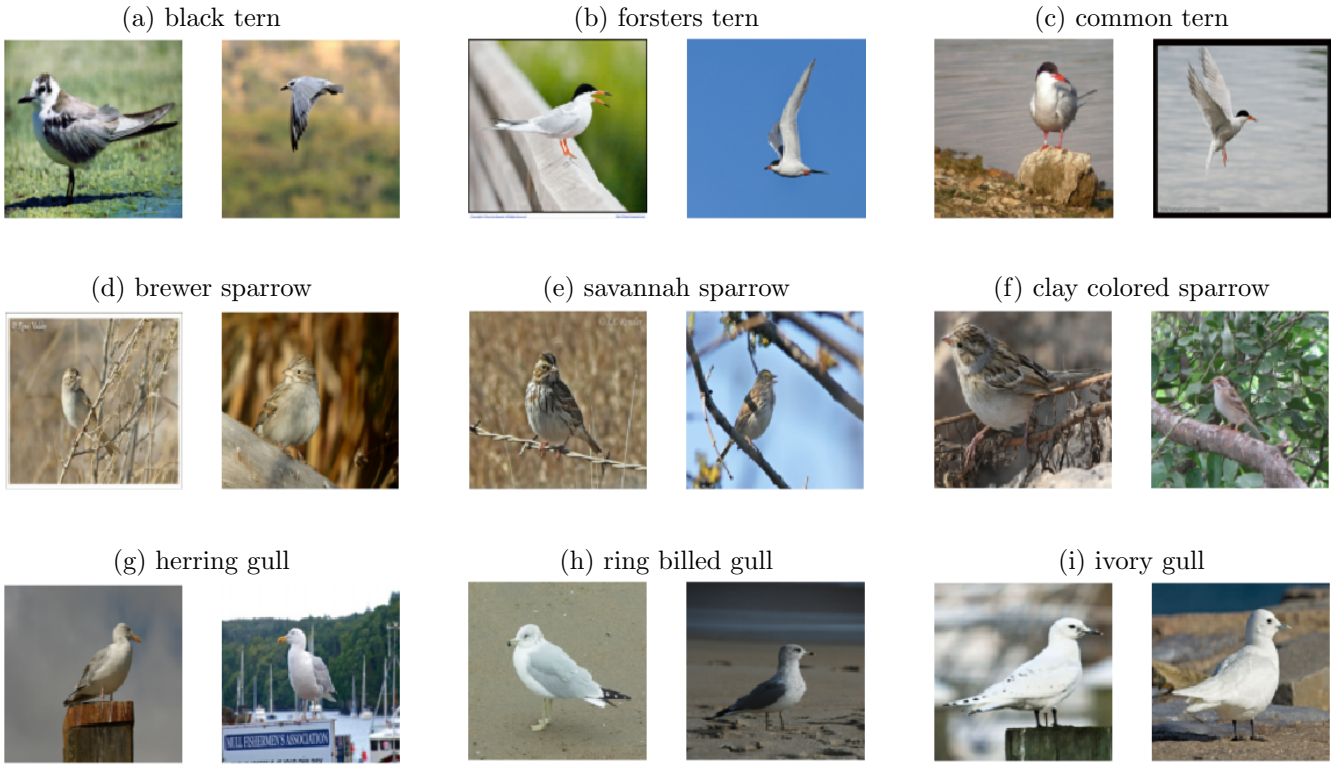


Figure 4: Example images of some bird families and species selected for this experiment. Each row represents a family of birds, and within each row there are 2 examples from 3 different species in that family.

**Choosing a feature extractor** As a feature extractor, we evaluated multiple pre-trained vision models, available on Keras applications (`https://keras.io/api/applications/`). However, when evaluating the performance of a logistic regressor trained on the embedding from each model, we observed unsatisfactory accuracies, in the $0.45 - 0.6$ range (for a 3-class problem). Building on the method presented in [5], we observed that concatenating image embeddings from few pre-trained models to a long vector of embeddings, and use that vector for classification improved the accuracy. In order to choose which models to use for concatenation, we evaluated the accuracy on all possible compositions of 3 & 4 models from a pool of 8 models, and choose the composition with highest accuracy. The models composition and embedding dimension reported in table 13.

| | |
|---|---|
| number of experiments in each dataset | 30 |
| experiment train\test sizes | 20\10 samples per class |
| SVM kernel grid | {RBF, polynomial} |
| SVM/LR regularization grid | {1e-3,1e-2,1e-1,1,1e1,1e2} |
| image embedding | concatenation of embeddings from:<br>EfficientNetV2B3 ([30])<br>MobileNetV2 ([28])<br>Xception ([4])<br>InceptionV3 ([29]) |
| concatenated embeddings dimension | 6912 |

Table 13: Experiments parameters for image classification using transfer learning.

Overall we report results for 5 classification tasks, each with 3 classes. On each task we embedded the images as described above, and then trained and tested simple classifiers (SVM, L1 regularized LR and also our method) on the embeddings. For each task we performed 30 experiments (different splits of the data). The results are presented in table 14. For each classifier we report the mean accuracy and the mean number of features selected. Overall the results are mixed, with our method (HC-ANOVA) always better than the HC-OvA approach, but is equivalent or slightly outperformed by either the SVM or logistic regression. Note however that HC-ANOVA based classifier achieve these results with considerably less features, and without hyper-parameters tuning.

**Assessing problem sparsity and ANOVA usefulness with a feature "oracle"** In every task in table 14, we used the HC-ANOVA method to first find a set of useful features using all the data (train+test), and then used this features knowledge to train and test another CS classifier (which we named "CS+HC-ANOVA Features Oracle") in all experiments of the task. It's important to stress that the oracle classifier was exposed only to the identity of useful features, but not to the test data itself (i.e., it wasn't trained on the test data). The mean accuracy and number of selected features for the oracle classifier are reported in table 14 as well, and from these we draw a few conclusions:

1. The number of features identified by the oracle is larger than what we would consider for a truly sparse problem (except from the "tern" task). Considering that HC-ANOVA selection tends towards over-selectivity (lower recall, higher precision in feature selection - see section 4.1), and assuming the features found are indeed useful (judging by the high accuracy), then the number of selected features is a rough lower bound to the true number of useful features. This means that $\epsilon \approx 0.02 \rightarrow 0.05$, corresponding to $\beta \approx 0.35 \rightarrow 0.4$ - not a sparsity level for which we saw true advantage of our method (section 4.1).

2. The high accuracy achieved by a simple (centroid similarity) classifier with the "oracle" features show that useful features can be detected by ANOVA, and indeed even a very simple classifier can be very accurate when fed with good features (at least in these problems). It is interesting and an open question whether the oracle features cannot be inferred from train data alone, or is it a question of feature selector design, and the HC-ANOVA can be improved to capture them.

| Bird Family | CS + HC-ANOVA | CS + HC-OvA | SVM | LR (L1) | CS + HC-ANOVA Features Oracle |
|---|---|---|---|---|---|
| tern (black tern, common tern, forsters tern) | **0.55** | 0.52 | 0.52 | 0.54 | 0.72 |
| | (2.23%) | (36.43%) | (100%) | (8.85%) | (0.93%) |
| sparrow (brewer sparrow, clay colored sparrow, savannah sparrow) | 0.56 | 0.52 | 0.52 | **0.585** | 0.77 |
| | (2.11%) | (15.36%) | (100%) | (6.89%) | (1.71%) |
| flycatcher (yellow bellied flycatcher, least flycatcher, olive sided flycatcher) | **0.62** | 0.60 | 0.60 | 0.61 | 0.82 |
| | (2.71%) | (77.98%) | (100%) | (8.43%) | (5.28%) |
| wren (carolina wren, house wren, winter wren) | 0.7 | 0.68 | **0.73** | 0.70 | 0.88 |
| | (2.62%) | (6.16%) | (100%) | (10.11%) | (3.14%) |
| gull (herring gull, ivory gull, ring billed gull) | 0.64 | 0.61 | **0.67** | **0.67** | 0.83 |
| | (2.72%) | (6.64%) | (100%) | (9.45%) | (4.5%) |

Table 14: Results for the 3-class bird species classification problems demonstrating small number of informative features. In each task, mean accuracies are in the top row (best in bold), and mean percentage of selected features (in parenthesis) in the bottom row.

# 6 Summary

In this project we extended the method developed by Donoho & Jin ([9]) to the multi-class case, utilizing the parametric/non-parametric ANOVA test p-values to score features instead of the original t-test p-values. We showed that in rare-weak data, this method ,when coupled with a simple classifier, is not only superior to a second, more immediate extension of [9], (the one-vs-all (OvA) approach), but also to a whole range of common classifiers.

Our classification experiments on gene expression data suggest that this method can be competitive for tabular, high dimensional data, especially when the data is scarce and use of modern, sophisticated classifiers is limited. The HC-ANOVA has other advantages as well: first, it is parameter-less and adaptive, and does not require tuning. Second, it is interpretable, providing statistically sound importance score for the features it detects. Third, our experiments on applying it to text and image embeddings showed that even when it is not the best method (accuracy wise), it strikes a good balance between performance and computational efficiency - i.e., may achieve nearly as good performance using number of features considerably smaller then alternative methods. An example application which may benefit from it is Retrieval-Augmented Generation (RAG), where trade-offs between embedding size, performance, and efficiency in retrieval tasks is often an issue ([12]), and this may be a direction for future research.

Although we didn't show clear performance advantage in any of the transfer learning use cases we tried, we believe the problems we tested were simply not sparse enough for the HC-ANOVA method to have

any advantage. However, with the advent of large language and vision models, they are becoming ever more parameter rich and general, and might produce sparse problems when used for domain adaptation. Further research is needed to find if and under what conditions the HC-ANOVA can be useful in these use cases.

## Code

The code for all feature selectors implemented in this project, as well as Jupyter notebooks with the experiments code, are available in the project Github repository: `https://github.com/roei-go/HC-feature-selection.git`.

## References


[1] R. Bommasani, D. A. Hudson, A. Adcock, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. URL `https://arxiv.org/abs/2108.07258`.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[4] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1251–1258, 2017.

[5] A. Chowdhury, M. Jiang, S. Chaudhuri, and C. Jermaine. Few-shot image classification: Just use a library of pre-trained feature extractors and a simple classifier. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9425–9434, 2021. doi: 10.1109/ICCV48922.2021.00931.

[6] D. Donoho and J. Jin. Feature selection by higher criticism thresholding achieves the optimal phase diagram. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367 (1906):4449–4470, 2009.

[7] D. Donoho and J. Jin. Higher criticism for large-scale inference, especially for rare and weak effects. 2015.

[8] D. L. Donoho and J. Jin. Higher criticism for detecting sparse heterogeneous mixtures. *The Annals of Statistics*, 32(3):962–994, 2004.

[9] D. L. Donoho and J. Jin. Higher criticism thresholding: Optimal feature selection when useful features are rare and weak. *Proceedings of the National Academy of Sciences*, 105(39):14790–14795, 2008.

[10] B. C. Feltes, E. B. Chandelier, B. I. Grisci, and M. Dorn. Cumida: An extensively curated microarray database for benchmarking and testing of machine learning approaches in cancer research. *Journal of Computational Biology*, 26(4):376–386, 2019. doi: 10.1089/cmb.2018.0238. URL `https://doi.org/10.1089/cmb.2018.0238`. PMID: 30789283.

[11] J. D. Gibbons and S. Chakraborti. *Nonparametric statistical inference*. CRC press, 2011.

[12] J. Guo, Y. Zhang, Y. Fan, and X. Cheng. Efficient retrieval-based neural network training for sparse and dense text representations. *arXiv preprint arXiv:2004.12578*, 2020. URL `https://arxiv.org/abs/2004.12578`.

[13] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar):1157–1182, 2003.

[14] M. Huh, P. Agrawal, and A. A. Efros. A closer look at transfer learning and domain adaptation in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 537–544, 2016.

[15] Y. I. Ingster, C. Pouet, and A. B. Tsybakov. Classification of sparse high-dimensional vectors. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1906):4427–4448, 2009.

[16] A. Khosla, A. Handa, D. Jayaraman, A. Gupta, and L. Fei-Fei. Novel dataset for fine-grained image categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3507–3513, 2011.

[17] S. Kornblith, J. Shlens, and Q. V. Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2661–2671, 2019.

[18] J. Krause, Y. Jia, and J. Deng. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 106–113, 2013.

[19] W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[20] P. Lewis, E. Perez, A. Piktus, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[21] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, pages 722–729, 2008.

[22] OpenAI. Introducing text-embedding-ada-002, 2022. URL `https://openai.com/blog/new-and-improved-embedding-model`.

[23] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8): 1226–1238, 2005.

[24] M. Pirooznia, J. Y. Yang, M. Yang, and Y. Deng. Comparative analysis of different machine learning methods on microarray gene expression data. *BMC Genomics*, 9(1):1–13, 2008.

[25] A. Radford, J. Wu, R. Child, et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:9, 2019.

[26] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 806–813, 2014.

[27] J. A. Rice. *Mathematical statistics and data analysis*. Cengage Learning, 2006.

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[30] M. Tan and Q. V. Le. Efficientnetv2: Smaller models and faster training. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 10096–10106. PMLR, 2021.

[31] C. Wah, H. Adam, M. Sadeghi, A. Lapedriza, W. Shih, P. Lazzari, and F.-F. Li. Caltech-ucsd birds 200: A benchmark for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2461–2468, 2011.

[32] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, 2016. doi: 10.1186/s40537-016-0043-6.