On the Subject of Identifying Anomalies in Heterogeneous Vector Based Data

Cyrus Cousins

Donna Slonim, Advisor

Undergraduate Thesis

Tufts University

Department of Computer Science

Spring 2015

Abstract

This paper is focused novel techniques for the unsupervised anomaly detection algorithm over datasets of real and finite discrete variables, centered around the existing FRaC algorithm. Novel variants of the FRaC algorithm are presented, alongside mathematical justification and empirical evidence to support their use. mFRaC, eFRaC and cFRaC, are introduced here. These techniques have a focus on ensemblification, treating features equally, and handling missing values in samples. eFRaC and cFRaC are shown to be minor improvements on the traditional FRaC algorithm over UCI datasets by comparison of AUROC values.

Additionally, several previously unknown properties of the FRaC algorithm are analyzed. The property of normalization-invariance is shown, given lenient assumptions. The properties of strong and weak self selection in feature modelling techniques are introduced, and FRaC is sown to have the weak self selection property under certain circumstances. Implementation details of the various statistical calculations necessitated by FRaC are also discussed.

Original filter method based heuristic feature selection techniques are presented, alongside analysis and empirical evidence. A more conservative FRaC specific alternative to traditional filtering, termed partial filtering, is also introduced, and compared to traditional filtering.

Finally, these algorithms are discussed in the context of the larger subfield of feature modelling anomaly detection techniques. The algorithms presented are broken into two categories, mathematical redefinitions of what constitutes an anomaly, and hyperparameter-reducing generalization algorithms. Additionally, further algorithms that were conceived but not implemented are described in this framework.

Keywords: *Data Mining* · *Machine Learning* · *Anomaly Detection* · *Semisupervised Learning* · *Feature Modelling* · *Probability* · *Information Theory* · *Filter Method*

Contents

List of Figures

List of Tables

List of Equations

Chapter 1: Introduction

1.1 Anomaly Detection

1.1.1 What is Anomaly Detection

Anomaly detection, in its most general form, is the problem of taking a set of *samples*, and identifying samples that are *different* from the others. In that definition, I used two very imprecise words, *samples*, and *different*. *Samples* can be made precise to fit any type of data; popular categories of data include text, time series information, or vectors of real, complex, and/or discrete variables. On the other hand, the word *different* evades precise definition, and is a large part of the reason why anomaly detection is such a complex topic.

Many anomaly detection methods calculate an outlier score for each point, and different algorithms for calculating outlier scores can be thought of as different definitions of *different*, so by their own definition of different, each non-heuristic outlier detection algorithm is supreme. Some techniques, rather than quantifying anomality numerically, produce a binary label of either *normal* or *anomalous*. This is convenient, because the results are easy to interpret, however knowing the degree of anomality is often desirable. This paper primarily focuses on the former type, because most algorithms are capable of outputting scores rather than labels, and it enables more sophisticated evaluation of an algorithm than does a binary label.

Because of the imprecise nature of an anomaly, anomaly detection algorithms are usually evaluated empirically over real world datasets. Both false positive and false negative rates are important in the evaluation of an anomaly, however, any discussion of false negative and false positive rates requires a firm classification of a sample as either “anomalous” or “normal”, and can thus be tuned in either direction by adjusting the size of those two groups.

An evaluation technique that operates on rankings, rather than on binary classifications, is the *area under the curve* of the *receiver operator curve* (AUROC) (Spackman, 1989). The AUROC is a valuable tool, because it summarizes the full range of information present in the set of outlier scores generated by applying an anomaly detection algorithm to a query set without requiring hard categorizations, and it is not sensitive to the relative proportion of normal to anomalous samples in the test set. The AUROC be interpreted as the probability that in some set of instances, a classifier will rank a randomly selected anomalous instance as more anomalous than a randomly selected normal instance (Fawcett, 2006).

Most of this paper is focused on variants of the FRaC algorithm, introduced in 1.3.4, which performs anomaly detection on vectors of real and discrete (alternatively referred to as nominal) variables. Many anomaly detection techniques are restricted to working solely on either real valued or discrete features, but more algorithms that work on heterogeneous vector data are also common.

1.1.2 Types of Anomaly Detection Problem

Supervised Anomaly Detection

The *supervised anomaly detection problem* is the problem of differentiating between a set of common *normal* classes and a set of uncommon *anomalous* classes, given instances of both as training data. The supervised anomaly detection is similar to supervised classification under a class imbalance (Chandola et al., 2009), because it can be considered a binary classification problem where the task is to distinguish between members of the *normal* and *anomalous* classes. Many classic techniques in the machine learning literature deal with class imbalance, and they largely fall into two major categories: cost function and sample based techniques.

Cost function based approaches work by optimizing a function of accuracy of correct and incorrect predictions for each class. Generally they assign high cost to misclassifying anomalies so as to obtain high recall. On the other end of the spectrum, sampling based techniques, such as AdaBoost (Freund and Schapire, 1997), SMOTE (Chawla et al., 2002), SMOTEBoost (Chawla et al., 2003), and RUSBoost (Seiffert et al., 2010) deal with class imbalance by over-representing minority class instances in training data, so as to train classifiers that are more likely to report them. Both of these categories deal with the class imbalance problem, and are thus applicable to supervised anomaly detection.

Like many semisupervised and unsupervised anomaly detection algorithms, FRaC is designed to operate without labeled anomalous instances, and does not benefit from this information. For this reason, supervised anomaly detection is not further discussed in this paper.

Semisupervised Anomaly Detection

The *semisupervised anomaly detection problem* is the problem of, given "normal" training data (containing no anomalies), building a model that can detect anomalous samples. The semisupervised anomaly detection is a harder problem than the supervised, because we have essentially removed the anomalous instances from the training data. Thus semisupervised techniques need to detect anomalies *without* an explicit concept of what anomalies look like.

Although it is harder problem, semisupervised learning is a very practical because it does not require labeling a dataset representative of anomalies. Furthermore, semisupervised techniques can identify novel categories of anomaly, which in a sense makes them more useful than supervised algorithms, as the latter can only identify anomalies that resemble those present in the training data. This is key, as in many fields and obtaining a dataset encompassing *all possible types* of anomaly is infeasible or impossible.

Semisupervised learning requires the curation of a set of normal behavior, which is potentially costly, however it is generally far easier to generate normal training data than it is to identify a sufficient number of anomalies to train a supervised algorithm.

Because semisupervised algorithms only have normal data available, they generally are trained by, either implicitly or explicitly, creating a model of what normal data looks like. Identifying anomalies (or scoring new instances with respect to an outlier score) then becomes a matter of finding how much an instance deviates from the learnt model.

Popular semisupervised algorithms include One-Class SVMs (Schölkopf et al., 2000, 2001), ADBag (Pevnỳ, 2013), Cross Feature Analysis (CFA) (Huang et al., 2003), FRaC (Noto et al., 2010, 2012). A myriad of traditional statistical techniques for identifying outliers may also be used in a semisupervised manner.

Unsupervised Anomaly Detection

The *unsupervised anomaly detection problem* is the problem of detecting anomalous samples that are mixed into a pool consisting mostly of normal samples. The unsupervised anomaly detection problem is a more difficult problem than its semisupervised analogue, because we do not have a way to examine a normal population in the absence of anomalous samples.

Note that by mixing the training and query data of an instance of the semisupervised problem, we can see that the semisupervised problem actually reduces to the unsupervised problem (furthermore the supervised problem reduces to the semisupervised, by removing the anomalous training instances, completing the reduction hierarchy between the three). Conversely, semisupervised anomaly detection algorithms can often be run in an unsupervised manner by training them on a subset of data, and assuming the subset contains sufficiently few anomalies so as not to render whatever model of normalcy the algorithm builds insensitive to further anomalies (Chandola et al., 2009). The efficacy of this technique depends on the dataset, the proportion of anomalous samples in the data, and the algorithm in question.

Popular unsupervised anomaly detection techniques include Local Outlier Factor (LOF) (Breunig et al., 2000) and K-nearest neighbors (not to be confused with the classification/regression algorithm of the same name). Many statistical and clustering based techniques are also directly applicable to the unsupervised anomaly detection problem.

1.1.3 Types of Anomaly Detection Algorithm

Distance Based Methods

Many of the early anomaly detection techniques fit into the distance based category, though due to the inherently simple hypothesi of distance based techniques and susceptibility to noise in high dimensional spaces, these techniques are less popular today. Statistical definitions of outliers are often distance based, and algorithms to find them can be considered distance based anomaly detection techniques.

A statistical outlier can be defined as "[an observation] that appears to deviate markedly from other members of the sample in which it occurs" (Grubbs, 1969). Simply taking the distance to the nearest point in the space is a trivial form of distance based method, though it is not at all resistant to sampling artifacts. A more advanced technique takes the distance to the k_{th} nearest point as the outlier score; this gives some resiliency to sampling artifacts and is an overall more robust metric, but it introduces a new hyperparameter, k, that is quite difficult to set. Knorr et al. discuss this and similar techniques in (Knorr and Ng, 1999).

Statistical definitions, such as Peirce's criterion (Peirce, 1852) and Chauvenet's criterion (Chauvenet, 1863) relate the probability of the occurrence of a sample to its anomality. These techniques are at heart essentially distance based methods, as they suffer from the same drawbacks, they divide spaces in a similar manner, and they find points that are distant from others.

Although the history of distance based techniques is a long one, more advanced computational techniques have long since dominated the field. The inability of distance based methods to cope with more diverse types of data, and the normality assumptions often made in statistical techniques are largely unsuitable for modern anomaly detectors. Furthermore, no distance based technique is invariant to transforms more complicated than scalar multiplication (uniform scaling) or constant sum (shifting), and robustness against complicated nonlinear transformations of data is necessary for many real world applications.

Density Based Methods

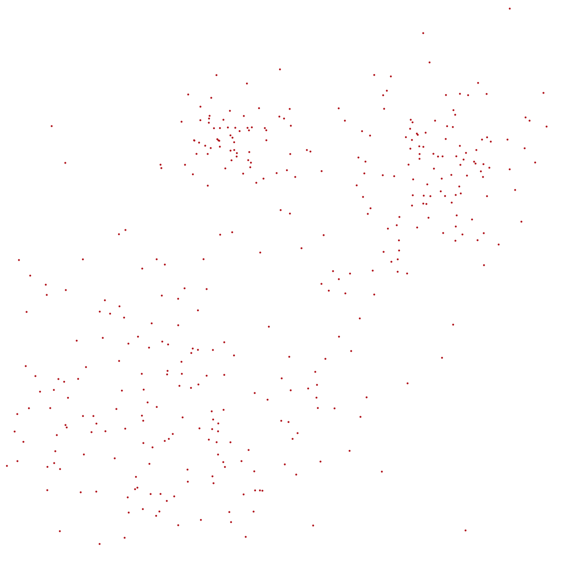
The canonical approach to density based anomaly detection is the *Local Outlier Factor* (LOF) technique, introduced in (Breunig et al., 2000). LOF is density insensitive, and, when properly tuned and on appropriate datasets, it finds points outside of *any* distribution in a mixture, regardless of the variance of the distributions in play. See Figure 1.1 for an example of the sorts of dataset on which LOF outperforms distance based techniques.

LOF does have the downside of introducing a tricky hyperparameter k: lower values increase vulnerability to small sample sizes, and higher values render the algorithm largely useless, as it begins to examine irrelevant points. In (Breunig et al., 2000), the authors suggest to sum the outlier scores produced by running LOF with a range of k values to mitigate these effects.

Both distance and density based techniques rely on calculating the distances between samples. When all features in a sample are real values, this can trivially be done with Euclidean distance, though other distance metrics such as L_1 distance are popular as well. In high dimensional spaces, noise can become dominant and issues associated with the *curse of dimensionality* (Bellman, 1957) can dominate, so both density and distance based techniques tend to work better in low dimensional spaces.

Feature Modelling Methods

Feature modelling techniques are a relatively new group of anomaly detection algorithm that operate by learning patterns in the relationships between features and identifying samples that



Density based techniques are capable of identifying anomalies regardless of the density of the distribution from which they originate. In this figure points are drawn nonuniformly from one of three normal distributions of different variances. Distance based techniques cannot distinguish the fringes of larger and sparser distributions from points not clearly associated with any distribution (particularly visible in the lower left distribution), whereas density based techniques generally will not consider such points to be anomalies.

Figure 1.1: Distance vs. Density Based Techniques

break these patterns as anomalous.

Feature modelling techniques are very general in scope, and can identify diverse types of anomaly. They operate in a manner quite unlike distance and density based techniques, but, with assumptions about hyperparameterization, are able to quite easily learn spaces on which distance and density based techniques perform well. Feature modelling approaches generally suffer from high training times, because a supervised learner must be trained for each of the features the algorithm models (which is usually all of them).

Cross Feature Analysis (CFA) (Huang et al., 2003) is the original feature modelling algorithm, though it has largely been superseded by later developments. CFA uses probabilistic predictors, and identifies anomalies based how poor the predictions are. FRaC (Noto et al., 2010, 2012) is a related algorithm that expands upon CFA in many ways, and calculates a very different anomaly score, based on surprisal rather than average probabilities. In (Tenenboim-Chekina et al., 2013), the authors describe two additional techniques, Feature Chains (abbreviated FC, so called because of the influence of the Classifier Chains multilabel classification technique (Read et al., 2011)), and Ensemble of Feature Chains (EFC), both based on CFA. These algorithms calculate an outlier score using conditional probabilities based on classifier chains, rather than the simple summation of CFA.

Ensemble Methods

There is a great breadth of anomaly detection techniques, and as discussed in Section 1.1.1, there is no single mathematically precise concept of what an anomaly is. A popular way to catch diverse anomalies in a single net is to use an *anomaly detection ensemble*. These techniques generally work by summing or otherwise combining several outlier scores produced by different anomaly detectors for a single sample. When eclectic anomaly detectors with vastly different concept of an anomaly are used, the resulting ensemble can detect anomalies of various types that no one method would identify.

Both FRaC and CFA are ensemble based techniques, as they combine the anomaly scores produced by many different models, however the true power of anomaly detection ensembles lies in the combination of vastly different techniques. Zimek et al. treat the subject of ensemble anomaly detection in great detail, and introduce further techniques, in (Zimek et al., 2014b,a). Ensemblification of supervised learners is discussed further in Section 1.2.3, and much of the theory of supervised learner ensembles holds for anomaly detector ensembles as well.

1.2 Relevant Background and Definitions

1.2.1 Probability Distributions

Discrete Probability Distributions

A discrete probability distribution is a set of (Category, Probability) pairs, where each category is unique, each probability is a real number in $[0, 1]$, and the probabilities sum to 1. All distributions used in FRaC additionally have the distinction of being finite. One may also think of a probability distribution as a function of categories onto probability values.

Support The *support* of a probability distribution is defined as the set of values for which the distribution assigns nonzero probability. It is notated $\text{supp}(X)$.

Estimation through Sampling Given a set of samples from some finite distribution, we can estimate the original distribution by counting the number of samples from each class, and dividing each count by the total number of samples. In FRaC, we often apply Laplace smoothing (often referred to in the FRaC literature as a pseudocount of one), where each possibility in the distribution is artificially added to the actual samples, to preclude attempts to take the surprisal of a zero probability event. This technique also simplifies entropy calculations, and probability theory suggests that it makes distributions less prone to sampling noise.

This technique of estimation through sampling with uniform prior is very much in line with the Frequentist interpretation of probability. The technique of estimating a true finite discrete distribution from a finite number of samples in this manner comes from Laplace's writings on the sunrise problem (Laplace, 1814), in which he estimates the probability with which the sun will rise tomorrow and explores the effects of evidence and prior knowledge. When this technique is used, for some number of samples n, a distribution X_n created by sampling from the true distribution X_T n times with Laplace smoothing has the following properties:

$$\begin{gathered}\forall a \in \text{supp}(X_T) \lim_{n\to\infty} X_n(a) = X_T(a) \\ \forall m, n : n \geq m, \\ \text{E}\left[\text{D}_{\text{KL}}(X_n \,\|\, X_T)\right] \leq \text{E}\left[\text{D}_{\text{KL}}(X_m \,\|\, X_T)\right]\end{gathered} \tag{1.1}$$

Although in practice we usually have a finite sample, this convergence provides theoretical justification for estimating a distribution through sampling, and the second equation (where $\text{D}_{\text{KL}}(A \,\|\, B)$ is a function of two distributions valued by their Kullback Leibler divergence, see Section 1.2.2) basically states that distributions made with more samples are likely to be closer to the true distribution than those made with fewer samples.

Continuous Probability Distributions

A continuous probability distribution (PDF) is a probability distribution over $\mathbb{R}$. A continuous probability distribution is represented with a continuous probability distribution function[1], which is a bounded nonnegative continuous function $f : \mathbb{R} \to \mathbb{R}_0^+$ such that $\int_{-\infty}^{\infty} f(x)\,\text{dx} = 1$.

Despite their many similarities, continuous distributions do not share all the properties of discrete distributions. Having a continuous distribution over $\mathbb{R}$, an uncountably infinite space, means that the probability of any individual event in the space must be 0 , and queries are generally made as to the probability of an event falling in some continuous *range*. For some continuous PDF f, the probability of an event falling between a, b where $a \leq b$ is given by $\int_a^b f(x)\,\text{dx}$.

[1] Note that functions *are* sets, and discrete distributions as I defined them *are* equivalent to functions. Generally, it is easier to think of discrete distributions as sets and continuous distributions as functions, so I refer to them as such.

Support The *support* of a PDF is defined as the set of values for which the PDF has nonzero value. It is notated supp(X).

Scaling a Continuous Probability Distribution The concept of *scaling* a probability distribution f by a nonzero real value a refers to horizontal scaling. Scaling a probability distribution is usually notated by subscript, so f scaled by a would be written f_a. We can define scaling in terms of the relationships between the scaled and unscaled distributions:

$$\forall x \in \mathbb{R} f(x) = a \cdot f_a(ax)$$

Another important property of the scale operation is that the result of scaling a probability distribution is itself a probability distribution.

Estimation through Sampling Unlike in the discrete case, calculating a probability distribution from a sample is a highly nuanced and complicated affair. In FRaC, Gaussian distributions and Gaussian mixture distribution functions are generally used to represent probability distribution functions. Fitting a sample to a Gaussian distribution is a straightforward and well defined process, but there are many ways to fit a sample to a Gaussian mixture function. This is further discussed in Section 1.2.4.

At times, one can perform calculations (such as expectation of a function) with either a probability distribution function or a sample of the function. This is usually a worthwhile pursuit, because converting a sample to a continuous distribution is an inherently lossy process, and can be computationally expensive. Furthermore, when the resulting distribution is integrated, more computational expense is incurred, making computations on samples of distributions an attractive option when possible.

1.2.2 Information Theory

Entropy is a numeric quantity which can be calculated on probability distributions. Introduced as an information theoretic concept (and giving rise to the field) in (Shannon, 1948), there are many interpretations, and several equivalent formulations. Thermodynamic Entropy is intimately related to Information Theoretic Entropy, though the latter is largely the focus of this paper.

Entropy is often described as the the average number of bits required to encode messages drawn from a probability distribution, but it may also be interpreted as the average surprisal when estimating a value drawn from a distribution X with a second value drawn from X. The *average surprisal* interpretation is the most prevalent in FRaC.

Surprisal and Entropy

Surprisal, also introduced in (Shannon, 1948), is defined on some probability p as:

$$\mathrm{I}(p) = -\log(p)$$

Surprisal may be interpreted as the amount of information in an event, and consequently is also referred to as *self-information*.

Choice of base rarely matters for the mathematics in this document, as long as it is consistent. Usually Euler's number (e) or 2 is used.

(Discrete) Entropy If entropy is defined as the average surprisal of a sample drawn from the distribution, then the formula for (discrete) entropy of some random variable X over the set $\vec{p}$ is as follows (with the addendum that elements in $\vec{p}$ not in the support of X do not contribute to the entropy of X):

$$\mathrm{H}(X) = \mathrm{E}[\mathrm{I}(X)] =$$
$$-\sum_{i=1}^{\dim \vec{p}} \begin{cases} 0 & \text{if } \mathbb{P}(X = \vec{p}_i) = 0 \text{, otherwise:} \\ \mathbb{P}(X = \vec{p}_i) \cdot \log \mathbb{P}(X = \vec{p}_i) \end{cases}$$

Differential Entropy *Differential Entropy* is a closely related concept to the discrete analogue, but despite the many similarities, many properties that hold for discrete entropy do not hold for differential. The differential entropy of some random variable X with PDF $f(x)$, and support

$\mathbb{X}$ also defined by Shannon in (Shannon, 1948), is defined as:

$$\mathrm{H}(X) = -\int_{\mathbb{X}} f(x) \log \big(f(x)\big)\, \mathrm{dx}$$

It is important to note that differential entropy differs from ordinary Shannon entropy in many ways. It is *not* the limit of the entropy of the discretized distribution as the number of categories goes to infinity (this is infinite), and it is not invariant under scaling. Despite having identical notation, it is important to use context and to not confuse differential entropy with ordinary Shannon entropy.

The entropy of a random variable X of some probability distribution scaled by some constant α is related to the entropy of the original distribution by the following formula:

$$\mathrm{H}(\alpha X) = \log(\alpha) + \mathrm{H}(X)$$

Divergence Functions

Divergence functions take two probability distributions onto a nonnegative real value, and generally signify how different the distributions are. Unlike distance functions, divergence functions need not be symmetrical or uphold the triangle inequality, however divergence functions *are* positive definite, so the divergence of two distributions is 0 if and only if the distributions are equal, and is positive in all other cases.

Kullback Leibler divergence (KL divergence), first described in (Kullback and Leibler, 1951), is a metric used to evaluate the efficacy of using an encoding optimized for one probability distribution to send signals from another.

The KL divergence of two discrete probability distributions P and Q is defined only when $\mathrm{supp}(P) \subseteq \mathrm{supp}(Q)$, and is given by:

$$\mathrm{D_{KL}}(P \,\|\, Q) = \sum_{i \in \mathrm{supp}(P)} P(i) \log \frac{P(i)}{Q(i)}$$

Similarly, the KL divergence of two continuous probability distributions f_P and f_Q is defined only when $\mathrm{supp}(f_P) \subseteq \mathrm{supp}(f_Q)$, and is given by:

$$\mathrm{D_{KL}}(f_P \,\|\, f_Q) = \int_{\mathrm{supp}(f_P)} f_P \log \frac{f_P(x)}{f_Q(x)}\, \mathrm{dx}$$

Note that KL divergence is invariant under scaling, meaning that $\forall \alpha \neq 0$, random variables X, $Y, \mathrm{D_{KL}}(X \,\|\, Y) = \mathrm{D_{KL}}(\alpha X \,\|\, \alpha Y)$.

KL divergence can be viewed as a measure of the similarity of two probability distributions, although it is not a true metric, because it is not symmetric. Because of this asymmetry, extra care must be taken to use KL divergence properly. Generally, it is useful to consider P or f_P to be the "true distribution", and the other distribution in question to be Q or f_Q, because then KL divergence can be interpreted as a measure of how much a distribution varies from the true distribution.

Alternatively, when no such distinction exists between two distributions, *symmetric KL divergence* can be used. Symmetric KL divergence is more difficult to interpret simply from an information theory perspective, but is of course symmetric. Symmetric KL divergence is defined as follows:

$$\mathrm{D_{KL^\circ}}(P \,\|\, Q) = \mathrm{D_{KL}}(P \,\|\, Q) + \mathrm{D_{KL}}(Q \,\|\, P)$$

Because of its symmetry, symmetric KL divergence can be used without worrying about interpreting P and Q, which is both a strength and a weakness. Due to the properties and restrictions of ordinary KL divergence, symmetric divergence is only defined when $\mathrm{supp}\, P = \mathrm{supp}\, Q$, and furthermore, note that it is also nonnegative.

Similar to the symmetric KL divergence, but with a different interpretation and fewer limitations are the class of λ divergences. Of particular interest is the Jensen-Shannon divergence introduced in (Lin, 1991), which is defined as

$$\mathrm{D_{JS}}(P \,\|\, Q) = \frac{\mathrm{D_{KL}}(P \,\|\, \frac{P+Q}{2}) + \mathrm{D_{KL}}(Q \,\|\, \frac{P+Q}{2})}{2}$$

Note that unlike the symmetric KL divergence, the Jensen-Shannon divergence makes no assumptions about the support of P and Q[2]. Furthermore, Jensen-Shannon divergence is confined to the interval of $[0, \log_2(a)]$ where a is the base of the logarithm used in the original formula.

1.2.3 Predictors

Predictor is a generic term for a regressor or a classifier. Predictors take labeled *training data* that contains both prediction features and a label to build a model that predicts labels from additional data. If the label is *discrete*, the predictor is a classifier, and if the label is *real*, the predictor is a regressor.

Predictor Ensemble

The benefits of ensemblification in machine learning are well known (Wolpert, 1992; Maclin and Opitz, 1999). According to (Dietterich, 2000), an ensemble classifier is more accurate than any of its individual components if and only if they are both *accurate* and *diverse*. In this context, *accurate* signifies accuracy greater than that of random guessing, and for two classifiers to be *diverse* means that they make different errors on the same data.

A traditional unweighted ensemble learner l works as follows: For training data $train = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \ldots, (\vec{x_n}, y_n)\}$, classifiers $c = \{c_1, c_2, \ldots, c_n\}$, l is trained by training each c on *train*, and a trained l is applied to a new instance $\vec{x}$ by applying some function f to a vector created by mapping each c over $\vec{x}$. For classifiers, f usually takes the classifier with the most votes, and for regressors f is usually *mean* or *median*.

A weighted learner uses weighted votes in classification, picking the class with the highest weight sum, and a weighted regressor takes a weighted mean. It is important to note that *the* highest weight sum is not guaranteed to exist, and the probability of a tie is particularly high in unweighted classification ensembles. This situation can be resolved in several ways. First, it can be resolved deterministically with a well ordered tiebreaker, though this can introduce biases, and alternatively, it can be resolved nondeterministically through random choice. The latter is used in all experiments presented in this paper.

Ensembles as Probabilistic Predictors Ensembles, weighted or not, are composed of supervised learners, but they are supervised learners themselves. In addition to being supervised learners, ensembles can be viewed as probabilistic predictors as well. The (possibly weighted) votes of a probabilistic classifier ensemble produce a vote distribution, which once normalized can be treated as a probabilistic classifier output.

Similarly, the votes of a regressor ensemble can be treated as samples of a continuous probability distribution (see Section 1.2.1), and they can be used as such or fit to a continuous probability distribution function.

1.2.4 Error Models

An *error model* is a model of the accuracy of a predictor. Error models are a form of conditional distribution, where both the evidence space and the distribution space are that of the predictor's output. When the predictor is a regressor, then these spaces are the reals, and when the predictor is a classifier, these spaces are finite discrete sets. Thus error models are continuous probability distributions, conditional on a real value, or finite discrete probability distributions, conditional on a discrete value.

An error model can be used to answer queries such as $\mathbb{P}(\text{True Class A} \mid \text{Predicted Class B})$, or $\mathbb{P}(\text{Predicted Value 6.28} \mid \text{True Value 2.71})$.

Note that in one case, the distribution is over true values, with an evidence space over predicted values, and in the other, the distribution is over predicted values, and the evidence space is over predicted values. Ordinarily, in FRaC, error models are queried with predicted values as evidence (to answer queries such as, for some discrete feature F, $\mathbb{P}(\text{Feature F is of true class B} \mid \text{Predicted class } A)$ for sample $\hat{\mathbf{x}}$), but at times it is useful to make queries on the predicted class using the true class as the evidence.

[2] This follows from the fact that $\text{supp}(\frac{P+Q}{2}) = \text{supp}(P) \cup \text{supp}(Q)$, and thus the support of $\frac{P+Q}{2}$ is a superset of the supports of both P and Q.

Different data structures can represent error models, such as matrices for categorical features, and various types of distribution valued functions of real numbers for continuous features. Some error models only support queries of one type (given one of either predictor values, or true values, but not the other), and other types support both types of queries.

Creating error models

Error models are generally built from a set of (prediction, truth) pairs, where the pairs are produced by pairing a truth value with a predictor's prediction for the truth value. Pairs can either be obtained from a *validation set*, as long as the elements of the validation set do not overlap the *training set*[3], or be obtained via cross validation (Stone, 1974) of the original training set.

When an error model is obtained through a validation set, it is more accurate to the model in question, because the error model is trained on the predictor itself. However, the predictor itself is generally less accurate, because some of the data that would otherwise be used to create it was withheld from the training set to form the validation set. Furthermore, if the validation set is small, the error model is subject to significantly more sampling bias than would be the case in cross validation. It is very important to consider these effects when creating an error model with a validation set.

On the other hand, when an error model is constructed through cross validation, it is less accurate, because the (prediction, truth) pairs used to construct it come from a collection of related models, all trained on different subsets of the training data. This effect is particularly devastating with unstable learners, such as decision trees or neural networks. However, all of the data can be used to build the predictor (after cross validation), and it can also all be used during cross validation to build the error model, without suffering the effects of constructing an error model on the training data. It is important to note that cross validation can be an extremely costly operation, as models need be trained, used, and destroyed repeatedly, whereas with a validation set, only a single model need be built.

For these reasons, when data are plentiful, a validation set is a better option, because it is both faster and more accurate. However, when data are limited, cross validation is more accurate, as more data can be used in the error model and the predictor. The decision of when to use cross validation and when to use a validation set is nontrivial, and entirely dependent on the dataset in question. When sufficient data exists, enough to build an accurate predictor is used for the training set, and if enough data remains for a validation set, the remainder of the data are used for that purpose, though determining when "sufficient" data are available is nontrivial.

Types of error models

Confusion Matrix Confusion matrices, known alternatively as error matrices, are in their most general form a way of keeping track of how many times instances of class A are classified as class B, for generic A, B. See (Stehman, 1997) for more information on using confusion matrices to measure classification accuracy.

Confusion matrices are built from pairs of (true class, predicted class) from a particular source, usually a single classifier, or in the case of cross validation, a set of classifiers generated in a similar manner. Usually, a confusion matrix is organized such that $\mathbf{M}_{A,B}$ is valued by the number of entries in the list of (true class, predicted class) pairs such that the true class is A and the predicted class in B. The convention of a vertical true class and horizontal predicted class will be used throughout this paper. Often Laplace Smoothing is used, in which case a constant, usually 1, is added to each cell of the confusion matrix. See Figure 1.2 for a comparison of different types of confusion matrix, alongside

[3] Strictly speaking, this statement is false. With predictor algorithms and error models with sufficiently simple hypothesis spaces to prevent overfitting, the training set and the validation set may overlap. These situations are quite nuanced, and depend heavily on structure of the dataset, so overlap is not acceptable in the general case.

Table 1.1: Overview of Error Model Types

Model Type	Query Directionality	Memory Cost	Query Cost	Model Specificity
Confusion Matrix	Bidirectional	$c^2 \in \theta(c^2)$	$\theta(c)$	Perfect
Normalized Confusion Matrix	Unidirectional	$c^2 \in \theta(c^2)$	$\theta(1)$	Perfect
Gaussian	Bidirectional	$2 \in \theta(1)$	$\theta(1)$	Normal Distributions Only, Error Distributions Only
Gaussian Mixture Distribution	Bidirectional	$2n \in \theta(n)$	$\theta(f(n))$	Error Distributions Only

In these analyses, c refers to the number of categories in a discrete variable, and n refers to the number of instances in the sample used to build the matrix. f refers to a function used to determine the number of Gaussian distributions in the Gaussian mixture distribution.
Here various concrete types of error models are presented. See Table 1.1 for a summary of these models.

Construction Set:

Pred	True
A	A
A	B
A	A
A	C

Pred	True
A	B
A	A
B	B
B	B

Pred	True
B	C
C	C
C	A
C	B

Confusion Matrix:

$$\begin{bmatrix} 3 & 2 & 1 \\ 0 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 3 & 2 \\ 1 & 3 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

Normalized Confusion Matrix:

True Class

$$\begin{bmatrix} \frac{4}{9} & \frac{3}{9} & \frac{2}{9} \\ \frac{1}{6} & \frac{3}{6} & \frac{2}{6} \\ \frac{2}{6} & \frac{2}{6} & \frac{2}{6} \end{bmatrix}$$

Predicted Class

$$\begin{bmatrix} \frac{4}{7} & \frac{3}{8} & \frac{2}{6} \\ \frac{1}{7} & \frac{3}{8} & \frac{2}{6} \\ \frac{2}{7} & \frac{2}{8} & \frac{2}{6} \end{bmatrix}$$

Figure 1.2: Various Forms of Confusion Matrix.

a comparative look at how they are built.

Normalized Confusion Matrix A *normalized confusion matrix* is a confusion matrix that has been processed so that each row (true class) or each column (predicted class) is a probability distribution.

In the normalization process, information is lost, notably the overall distribution of both the true and predicted classes (making conditional distribution calculations impossible), as well as distributions in the antidirection of normalization. Because this information is lost, a normalized confusion matrix can only be used to answer queries of the form $\mathbb{P}(\text{true} \mid \text{predicted})$ *or* $\mathbb{P}(\text{predicted} \mid \text{true})$, but *not* both.

See Figure 1.2 for an example of the construction of a normalized confusion matrix.

Gaussian A *Gaussian* function is a function of the form

$$ae^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

for constants $a, \mu, \sigma : a, \sigma > 0$ (Gauß, 1809).

It can be shown that Gaussian distributions are strictly positive continuous functions, μ is both the expected value and the maximum value, and σ is the standard deviation. Furthermore, when $a = \frac{1}{\sigma\sqrt{2\pi}}$, it can be shown that $\int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\,\mathrm{dx} = 1$, thus Gaussian distributions of this form are continuous probability distribution functions.

Error models need to function as *conditional* continuous probability distributions, so querying the likelihood of a model requires two arguments:

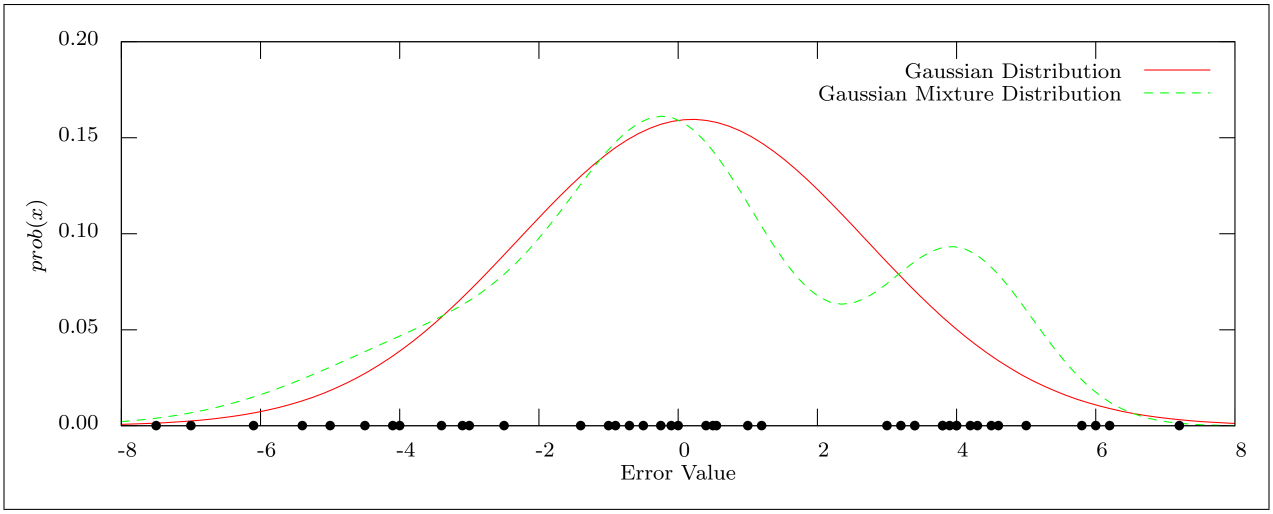


Figure 1.3: One Dimensional Continuous Error Models.

an evidence value and the value at which the query is directed (as in $\mathbb{P}(\text{value} \mid \text{evidence})$. To model this with a univariate function, we build a Gaussian of the *error* distribution, where error is defined like so:

$$\text{error}(t, p) = t - p$$

Where t denotes the true value and p the predicted value p of a feature. With this technique, a Gaussian function (or another univariate function) can represent the error distribution of a model, but situations where $p - t = k$ for some constant k (e.g. $p = 2, t = 4$ and $p = 0, t = 2$) can not be differentiated.

An example of a Gaussian distribution fit to error values is shown in Figure 1.3.

Gaussian Mixture Distribution Gaussian mixture distributions are functions composed of (possibly weighted) mixtures of Gaussian distributions. Error values are not always Gaussian distributed, and Gaussian mixture distributions have vastly larger hypothesis spaces than do Gaussian functions. See (John and Langley, 1995) for an overview of the advantages of a Gaussian mixture distribution for modelling probability distribution functions, and see figure 1.3 for an example of fitting a Gaussian mixture distribution to error values.

There are many ways to fit a Gaussian mixture distribution to a set of points. A simple technique used in FRaC is to build a histogram of equal bin sizes from the data, and mix together a Gaussian distribution for each bin, its weight that of the bin, its mean the center of the bin, and its standard deviation the width of the bin (John and Langley, 1995). As the number of bins increases, this converges to an arbitrary (not necessarily continuous) distribution, though a finite mixture of Gaussian distribution is necessarily continuous.

A similar technique is to partition the data into regions with equal numbers of samples, and mix equally Gaussian distributions centered at the center of each region, with standard deviation equal to the width of the bin.

More advanced techniques exist, but generally speaking, any technique that converges to the true distribution as the number of Gaussian distributions in the mixture increases is adequate to approximate continuous probability distributions.

Note that with Gaussian distributions g, $\forall x \in \mathbb{R}, g(x) > 0$, and similarly this holds for Gaussian mixture distributions. This property simplifies the calculation of many information theoretic constructs that would otherwise need guard against zeros. This is valuable from an implementation standpoint, and it also means that the KL divergence of any two Gaussian or Gaussian mixture distributions is defined.

1.3 FRaC

1.3.1 What is FRaC

In this section I introduce the FRaC algorithm and summarize the FRaC literature.

Feature Regression and Classification (FRaC) was introduced in the paper *Anomaly Detection Using an Ensemble of Feature Models* (Noto et al., 2010) as a semisupervised anomaly detection algorithm, and further expanded to work in the unsupervised case in *FRaC: a feature-modeling approach for semi-supervised and unsupervised anomaly detection* (Noto et al., 2012). FRaC is conceptually related to the earlier Cross Feature Analysis (CFA) algorithm, introduced in *Cross-feature Analysis for Detecting Ad-Hoc Routing Anomalies* (Huang et al., 2003), which works on a very similar principle.

The primary differences between FRaC and CFA are first that FRaC supports more types of features and predictors, and second that FRaC uses the information theoretic *normalized surprisal* outlier score to CFA's simpler probabilistic *Average Probability* score. Specifically, CFA supports only probabilistic classifiers of discrete values, whereas FRaC supports ordinary (non-probabilistic) classifiers and regressors of discrete and continuous values. A third distinguishing feature is that FRaC builds error model through a validation phase, whereas CFA relies on probabilistic classifiers, such as the Naïve Bayes, to determine its probabilities. Probabilistic classifiers generally make assumptions when determining probabilities, and when these assumptions are violated, their probabilities can be inaccurate. FRaC's error models generally make few assumptions, and are (usually) built through cross validation, which although susceptible to some bias, is unlikely to be sufficiently wrong so as to degrade performance.

1.3.2 Algorithm Definition

The FRaC algorithm (Noto et al., 2010, 2012) works by treating each feature in turn as a label, and training a classifier or regressor for said feature using the remaining features. It then constructs an error model for each predictor using a validation set (alternatively the error model is constructed by cross validating the training set).

After training, FRaC evaluates novel samples (the query set) by predicting each feature with each predictor, and summing the surprisal values of the conditional probabilities of each true value given the predicted value (these conditional error values are obtained by querying the error models). This outlier score is called *normalized surprisal*, and can be expressed mathematically as:

$$\mathrm{NS}(\hat{\mathbf{x}}) = \sum_{i=1}^{F}\sum_{m=1}^{M}\begin{cases} 0 \text{ if } \hat{\mathbf{x}}_i \text{ is missing, else:} \\ \mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i \mid C_{i,m}(\rho_i(\hat{\mathbf{x}})))\right) - \mathrm{H}(\hat{\mathbf{v}}_i^T) \end{cases} \tag{1.2}$$

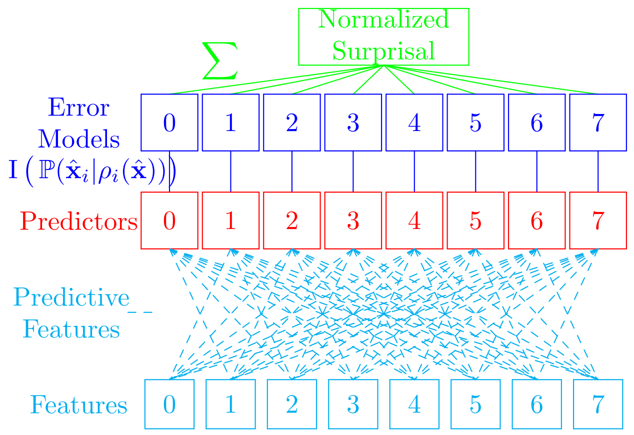
where $\hat{\mathbf{x}}$ is a query instance, $\hat{\mathbf{v}}$ the training set, F the number of features in the dataset, Mi the number of predictors for each feature, $C_{i,m}(\hat{\mathbf{x}})$ the prediction of the m^{th} predictor for feature i given $\hat{\mathbf{x}}$, and $\rho_i(\hat{\mathbf{x}})$ a vector identical to $\hat{\mathbf{x}}$ sans the i^{th} feature[4]. $\mathrm{H}(\widehat{\mathbf{v}_\mathbf{i}^\mathbf{T}})$ is perhaps obtuse notation for the entropy of the i^{th} feature in the training set. A graphical overview of the FRaC algorithm is provided in Figure 1.4.

Note that the double sum represents an anomaly detection ensemble, where each inner summand represents the output of a weak classifier. These inner summands are sometimes referred to as *partial normalized surprisal* or *featurewise normalized surprisal.*

1.3.3 Evaluation of FRaC in the Literature

FRaC has been shown to outperform other semisupervised anomaly detection algorithms (Noto et al., 2010, 2012), such as CFA (Huang et al., 2003), Local Outlier Factor (LOF)

[4] This formulation of normalized surprisal differs slightly from that of (Noto et al., 2012). The change is purely aesthetic.



All features, predictors, and error models are shown. A *predictive feature* line indicates that a predictor uses a feature to make predictions, and a *target feature* line indicates that a predictor predicts the target feature. A line from a predictor to an error model indicates that an error model calculates the surprisal of the true class given the predicted class, and lines from error models to the normalized surprisal box indicate summation of surprisal values to produce normalized surprisal values. *Subtraction of entropy values not shown.*

Figure 1.4: FRaC Data Dependency Diagram

Data Set	SVM Noto	LOF Noto	CFA	FraC Noto	FrAC Pevnỳ	PCA	KNN	LOF Pevnỳ	SVM Pevnỳ	ADBag
abalone	0.58	0.51	0.39	0.48	0.44	0.46	0.46	0.5	**0.6**	0.59
blood-transfusion	0.57	0.57	0.51	**0.59**	0.41	0.53	0.56	0.46	0.53	0.56
breast-cancer-wisconsin	0.51	0.93	0.3	**0.96**	0.94	**0.96**	0.95	0.95	0.9	**0.96**
ecoli	**0.98**	**0.98**	0.5	0.97	0.93	0.97	**0.98**	**0.98**	**0.98**	**0.98**
glass	0.61	**0.7**	0.64	0.65	0.64	0.68	0.67	0.68	0.55	0.67
haberman	**0.67**	0.64	**0.67**	**0.67**	**0.67**	0.61	**0.67**	0.66	0.33	0.64
ionosphere	0.84	0.91	0.87	**0.97**	0.96	0.96	**0.97**	0.95	0.93	0.96
iris	**1**	**1**	0.97	**1**	**1**	**1**	**1**	**1**	**1**	**1**
letter-recognition	0.99	0.99	**1**	**1**	0.99	0.99	0.98	0.98	0.79	0.96
libras	0.63	0.69	0.81	**0.89**	0.66	0.88	0.72	0.75	0.72	0.82
page-blocks	0.57	0.95	0.72	0.89	**0.96**	**0.96**	**0.96**	0.91	0.86	0.95
parkinsons	0.75	0.67	0.45	0.64	0.71	0.68	0.61	0.67	**0.85**	0.74
wine	0.79	0.88	0.33	**0.96**	0.93	0.95	0.95	0.93	0.91	0.93
yeast	0.69	0.72	0.56	0.72	0.72	0.72	0.72	0.71	0.67	**0.74**
Average	0.727	0.796	0.623	0.814	0.783	0.811	0.8	0.795	0.759	**0.821**
Maximal Count	3	3	2	**8**	3	3	5	2	4	5
Avg Rank	5.8	4.467	7.933	**3.067**	5.067	3.6	3.6	5	6.2	3.2

Table 1.2: Comparison of 10 semisupervised anomaly detection algorithms.
Data adapted from (Noto et al., 2012) and (Pevnỳ, 2013), only datasets common to both experiments were used.
Color in the table conveys no information beyond the values themselves, but is instead intended to convey intuition and direct attention: darker AUROC values represent higher AUROC values, darker discrete and real feature counts represent a dataset more imbalanced toward either category, and light training and test set colors represent abnormally small datasets.

Table 1.3: Correlations between AUROC scores on UCI datasets for semisupervised anomaly detection algorithms.
Data adapted from (Noto et al., 2012) and (Pevný, 2013), see Table 1.2. Color carries no additional information and is used solely to direct attention to high correlation values.

(Breunig et al., 2000), and One-Class SVMs (Schölkopf et al., 2000) in terms of accuracy over standard datasets, such as those available in the UCI repository (Lichman, 2013).

Were we to take these results in isolation, it would seem that FRaC is clearly dominant in the semisupervised anomaly detection problem. However, in the paper *Anomaly Detection by Bagging* (Pevný, 2013), Pevný finds that FRaC performs the second worst on the semisupervised anomaly detection over UCI datasets out of a PCA based anomaly detection technique (Shyu et al., 2003), k^{th} nearest distance (referred to, perhaps confusingly, as KNN), LOF, a one class SVM, and Pevný's own technique, ADBag (Pevný, 2013), though not by a large margin (average rank 4 out of 6), outperforming only the one class SVM. All learners performed relatively evenly overall in this experiment (average ranks all fell in $[3.1, 4.3]$), however it is interesting to note that FRaC does not perform as well on the supervised anomaly detection problem when evaluated by a third party.

I have two explanations for the inconsistencies between Pevný's experiments and those of the FRaC authors. First, FRaC is a complex meta algorithm with many hyperparameters; choice of learners and error models heavily influences the accuracy of FRaC. In Pevný's work, he used FRaC with only the simple linear least-square regression technique (Gauß, 1823), but as shown in (Noto et al., 2012), FRaC is far more accurate when used with an ensemble of regressors. In the Noto paper, a linear kernel SVM, a radial basis kernel SVM, and a decision tree were used. In some sense, the comparison in Pevný's paper is unfair, as although the ADBag algorithm is inherently far simpler, it is still an ensemble technique. In the experiments in question, ADBag was used with an ensemble of 150 weak anomaly detectors, against FRaC's weak collection of one linear least-square regressor per feature.

The data from both (Noto et al., 2012) and (Pevný, 2013) are presented in Table 1.2. Unfortunately, only 14 datasets were common to both papers, so the significance of their junction is debatable, and furthermore, the datasets underwent different processing, so the comparison is not entirely fair. We can however clearly see that Pevný's FRaC is inferior to the FRaC analyzed in Noto et al. in all but two datasets (`page-blocks` and `parkinsons`), whereas the relative performance of the common algorithms (one-class SVM and LOF) is similar. We also see that although ADBag does have the highest average AUROC, FRaC has the most datasets with maximal AUROC values and the lowest average rank.

In addition to the poorly tuned FRaC used in Pevný's work, FRaC may be represented overly well in (Noto et al., 2010) and (Noto et al., 2012), because FRaC is capable of operating over continuous and discrete variables, whereas the algorithms against which it competed operate only on continuous variables. Conversion from a discrete to continuous variable is a nearly lossless process (only context is lost), but it drastically increases the dimensionality and complexity of the space, so it stands to reason that an algorithm that is specialized to handle discrete variables may perform better than one that is not. FRaC's handling of missing variables also provides a slight advantage, although for the most part these were not common over the UCI datasets. Pevný's experiment operates only over "classification problems with numerical attributes without missing variables," so it affords

FRaC neither of these advantages.

Inconsistencies between the other algorithms are for the most part minor (note the correlations in Table 1.3). Noto's LOF is for the most part more accurate: Noto et al. use an ensemble over $k \in [10, 100]$ for LOF (Noto et al., 2012), whereas Pevnỳ simply uses a static $k = 10$ in his experiment (Pevnỳ, 2013), which explains the discrepancies between LOF AUROC scores between the papers. The two SVMs are vastly different, with a correlation of only 0.527, but one-class SVMs are complicated and have many hyperparameters. In this case, Pevnỳ's SVM used a Gaussian kernel, and that of Noto et al. used a radial basis kernel, so this discrepancy is not surprising.

In addition to its use as a standalone anomaly detector, FRaC is an integral part of the later CSAX (Characterizing Systemic Anomalies in eXpression data) algorithm, introduced in (Noto et al., 2014), for detecting anomalies in biological networks. CSAX is also an unsupervised anomaly detection that additionally takes a group of feature sets, and identifies anomalies based on regions of consistent anomality in these sets. CSAX has been used to analyze anomalies in RNA microarray data and characterize the anomalies in terms of the gene sets[5] involved.

1.3.4 Properties of FRaC

Scaling Invariance

An important but subtle property of FRaC is that, given reasonable assumptions about hyperparameterization (error model types and learning algorithms), normalized surprisal scores are invariant under scaling of continuous features.

Informally, the necessary assumption is that no learner or error model is affected by the scaling of any feature, be it the predictor and error model for said feature or the learnt models of other features. Firstly, this entails that for some feature f and scaled factor α, each learning algorithm for αf generates a model that is functionally identical to that generated for f, except with output scaled by α, on identical training data. Secondly, it also requires that learners for features other than f, when trained with αf, generate functionally identical models to those trained with f, except for scaling by α the input of f.

These criteria seem rather stringent, but algorithms that meet them or approximate them are in reality quite common. Note that decision trees generally do not make any sort of absolute quantity based decisions, and their rules will scale with the input features. Additionally, linear gradient descent techniques, like perceptrons, linear SVMs, and Neural Networks are for the most part resistant to scaling. However, distance based techniques, such as KNN are *very* sensitive to this type of scaling, and they thus violate the above assumption. It also requires that, for a unidimensional error model, the error distribution scales with the feature itself (i.e. scaling the feature by x produces an identical scaling in the error distribution).

Proposition 1. *Scaling-insensitive predictor hyperparameters imply scaling-insensitivity of FRaC.*

Proof. This proof shall proceed inductively. The inductive step shall be that scaling a particular feature in both the training and query set shall not change the normalized surprisal of an arbitrary element of the query set, and the base case shall be the trivial fact that the normalized surprisal of a sample in the query set is equal to the normalized surprisal of said sample, unscaled.

Now, suppose we have some continuous feature with PDF f and error distribution $\mathrm{Err}(f)$. We scale the feature by a, and call the scaled feature distribution f_a. The resulting error distribution, $\mathrm{Err}(f_a)$, by assumption has the property that $\mathrm{Err}(f_a)(ab) = \frac{\mathrm{Err}(f)(b)}{a}$. Now, we get that the normalized surprisal term for this feature, value x, in the original distribution with the original error model is $\mathrm{I}(x) - \mathrm{H}(f)$, and the normalized surprisal term for this feature, value

[5] *Gene sets* is an intentionally vague term: CSAX uses the Gene Set Expression Analysis (GSEA) algorithm, introduced in (Subramanian et al., 2005), to identify highly anomalous sets.

ax (which is just x scaled), in the scaled distribution with the new error model is $\mathrm{I}(ax) - \mathrm{H}(af)$. We now have:

$$\begin{aligned}
&-\log\big(\mathrm{Err}(f_a)(ax)\big) - \mathrm{H}(af)\\
&= -\log\Big(\frac{\mathrm{Err}(f)(x)}{a}\Big) - \big(\log(a) + \mathrm{H}(f)\big)\\
&= -\big(\log(\mathrm{Err}(f)(x)) - \log(a)\big) - \log(a) - \mathrm{H}(f)\\
&= -\log\big(\mathrm{Err}(f)(x)\big) + \log(a) - \log(a) - \mathrm{H}(f)\\
&= -\log\big(\mathrm{Err}(f)(x)\big) - \mathrm{H}(f)
\end{aligned}$$

We see that even after scaling a feature, the feature makes the same contribution to normalized surprisal. Because we also assumed that scaling a feature did not affect the other predictors, their contributions remain unchanged as well.

In the above mathematics, the possibility of missing features was neglected. If we define scaling a missing feature to produce a missing feature, then clearly scaling has no effect on missing features, as the normalized surprisal contribution of a missing feature is 0. Thus the normalized surprisal contribution of a feature is not changed under scaling, even if the feature is 0.

Now, by induction, I note that iterating this process and scaling each feature in turn results in scaled training and test sets such that a feature in the unscaled test set has the same normalized surprisal as a feature in the scaled set, thus we may conclude that normalized surprisal is invariant under scaling.

□

This seemingly innocuous result has several practical applications. Firstly, if the same assumptions are made, with similar assumptions about shifting of continuous variables, it follows that FRaC is z-score normalization insensitive. Furthermore, this result provides a theoretical justification for the scaling performed in eFRaC, discussed in Section 2.1.

Expected Normalized Surprisal of Normal Instances

In (Noto et al., 2012), Noto et al. write that on normalized surprisal, "[When] the query instance feature value $\hat{\mathbf{x}}_{qi}$ is missing, FRaC uses the entropy (this is the expected surprisal) of feature i's training set distribution in place of the surprisal score. This is equivalent to subtracting the entropy from each surprisal score and using zero when the feature value is missing." It is tempting to think that, under this second interpretation, for subsequent normal (non-anomalous) instances, we have a sum of the form:

$$\sum_{X \in \hat{Y}} \mathrm{I}(X) - \mathrm{E}[\mathrm{I}(X)]$$

and of course the expectation of this sum is 0. However, this does not follow; the fallacy stems from a misinterpretation of the "expected surprisal" interpretation of entropy.

Since the predictions for a feature are made by trained predictors, and not by sampling randomly from the feature distribution, it is reasonable to expect $\mathrm{I}(X)$ to be less than $\mathrm{H}(X)$, as long as the predictors are more accurate than randomly sampling from the feature distribution. Therefore, on average and depending on the predictability of the features in a dataset, normalized surprisal scores of normal instances are negative.

Under certain circumstances antagonistic to FRaC, the above does not hold. When a dataset consists solely of features that are not predictable from the remaining features, and models are learnt that make predictions with the same distribution as the true distribution, then normalized surprisal can be expected to be 0. This seemingly contrived scenario could presumably happen if all classifiers had prediction distributions equal to the true distributions, and all features are independent of one another.

A Data Centric Remedy Although expected normalized surprisal of normal samples is not 0, if an application requires that they are, there is a simple trick to attain this property.

The mean normalized surprisal values of the normal instances in the query set may be subtracted from each normalized surprisal value in the query set. This offsets each outlier score by a constant value, and thus does not affect rankings, but it does have the effect that the mean normalized surprisal value of the normal instances, and thus their expected value, is 0.

Alternatively, in the normalized surprisal formula, instead of subtracting entropy, we could subtract true expected surprisal. Expected surprisal may be calculated via cross validation or through a validation set, with the usual caveats about these techniques. This is more sophisticated than the previous technique, as it handles missing values more elegantly, but both result in expected normalized surprisal values of 0.

Self Selection Property

In this section, I define two concepts, the (strong) *self selection property* and the *weak self selection property*, specific to feature modelling techniques, explain their utility, and show conditions under which FRaC exhibits the weak self selection property and the strong self selection property. Both are related to the general idea of *introspection*, where a learning algorithm measures the accuracy of some component and incorporates this knowledge in subsequent models. Model selection by cross validation, as in (Zhang, 1993; Shao, 1993), is an early example of such introspection. Similarly, FRaC constructs error models and uses them to calculate surprisal values, so it too can be said to be introspective.

I first define the concept of a *poor predictor* to be one such that conditional distributions of true values for each predicted value are all equal, ignoring predicted values for which the resulting conditional distribution does not exist because it would be empty. A poor predictor must then be identified through its error value, and is thus dependent on the validation phase. Note that this definition implies that the set of conditional distributions of *predicted* values for each *true value* are all equal as well. Therefore, intuitively, a poor predictor is one that makes identically distributed predictions regardless of the evidence. Some features, such as those that take on only one value, will only produce poor predictors, and predictors of features that are independent of the remaining features converge to poor predictors as training and validation data tend to infinity, because the independence hypothesis makes the features entirely unpredictable. See Figure 1.5 for confusion matrices representing poor predictors.

I now define the self selection property of a feature modelling technique to be as follows: any poor predictor shall make no contribution to the outlier score. In the case of FRaC, where outlier scores are sums, this means that the surprisal of all predictions shall equal the entropy, and thus make a contribution of 0 to normalized surprisal. I similarly define the weak self selection property to be as follows: any poor predictor shall *on average* make no contribution to the outlier score, averaged over multiple samples. Thus in the case of FRaC (as in any feature modelling technique where feature scores are combined through summation), the termwise score must have an *expected value* of 0.

An algorithm with the self selection property essentially completely ignores the contribution of all poor predictors. This is intuitively a good thing, because if a feature can't be predicted, then predictions for the feature should have no effect on the anomaly score, because such predictions are meaningless. An algorithm with the weak self selection property allows such features to make contributions to the anomaly score as long as overall they do not bias it one way or the other. This seems like a useful property, but in reality it is only marginally useful, because even if the feature contributed a bias, it would contribute the bias systematically across the entire query set, and rankings would remain constant, even if absolute outlier scores did not. However, in FRaC, it does have the distinction of making the expected contribution of a missing and a present unpredictable feature identical (0), so it is still of some practical use.

I now show that if the prediction distribution

$$\begin{bmatrix}1 & 2 & 3\\1 & 2 & 3\\1 & 2 & 3\end{bmatrix} \quad \begin{bmatrix}1 & 2 & 3\\2 & 4 & 6\\1 & 2 & 3\end{bmatrix} \quad \begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\3 & 6 & 3\end{bmatrix} \quad \begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} \quad \begin{bmatrix}37 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix} \quad \begin{bmatrix}0 & 0 & 0\\0 & 1 & 2\\0 & 2 & 4\end{bmatrix}$$

Figure 1.5: Various Confusion Matrices Representing Poor Predictors

of each poor predictor matches the corresponding feature's true prediction (as is the case with some types of bias-free learners), then FRaC has the weak self selection property. For a given poor predictor, the expected contribution to normalized surprisal is 0, because either the feature is missing, in which case the contribution is 0, or the feature is present, but the expected surprisal is equal to the entropy. This statement follows from the definition of a poor predictor and the assumption, as the true distribution conditional on any predicted class (from which surprisal scores are drawn for normalized surprisal) equals the overall true distribution, and the expected surprisal of this conditional true distribution is therefore equal to the entropy of the overall true distribution), and thus the *expected* contribution for said feature is 0. In either case, the expected contribution is 0, so the overall expected contribution for the feature is 0. Because this applies to all poor predictors, it follows that FRaC has the weak self selection property.

If we now additionally assume that all features are uniformly distributed (in the normal class), it follows that FRaC has the strong self selection property. This is because for any uniformly distributed feature the conditional distribution of true values on any predicted value is also uniform, and thus *every prediction has identical surprisal.* It then follows that surprisal equals expected surprisal, and thus all normalized surprisal contributions are equal to 0. Therefore, all features are uniformly distributed, then all poor predictors make no contribution to normalized surprisal, and the strong self selection property is satisfied for FRaC.

We now have vocabulary for discussing the response of a feature-modelling technique to unpredictable features. Ideal algorithms have certain properties, and under certain circumstances, FRaC can have the weak self selection property and even the strong self selection property. This theoretical analysis suggests that under certain hyperparameterizations, such as those that produce identical predicted and true distributions, FRaC may have better performance. It also suggests that given such a hyperparameterization, FRaC is very resistant to difficult or unpredictable features when said features have uniform distributions.

Although FRaC does not in the general case have the self selection property, it is capable of introspection in a more general sense, in that FRaC trains error models in a validation phases, and bases decisions on them. In many practical contexts, because of this introspection, the contributions of poor predictors are outweighed by the contributions of better predictors, even when poor predictors vastly outnumber their superiors. In (Noto et al., 2012), an experiment was performed in which FRaC was able to detect anomalies with large number of additional randomly generated features. This experiment also shows FRaC's resilience against poor classifiers.

1.4 Contributions of This Paper

In this paper, theoretical analysis of traditional FRaC, as well as novel variants of FRaC, is presented. The first important contribution is the scaling insensitivity property above.

A technique to further correct for missing values, Missing value corrected FRaC (mFRaC), is presented in Section 2.3; a technique moving toward equipotent feature contributions, Entropy normalized FRaC (eFRaC), is presented in Section 2.1; and a technique to more effectively use ensembles of predictors in FRaC, Combinatorial ensemble FRaC (cFRaC), is presented in Section 2.2. These techniques refine the normalized surprisal anomaly score to be more effective at detecting anomalies in real world data.

In addition to the algorithmic variants presented here, heuristic techniques for feature selection by the filter method are discussed in Section 3.1. Somewhat surprisingly (due to the self selecting nature of FRaC), in some instances filtering is able to substantially raise the accuracy of FRaC. In addition to ordinary filtering, the FRaC-specific technique of partial filtering is also introduced in Section 3.2.1.

Finally, these techniques are categorized into refinements of the definition of an anomaly and generalizations of existing techniques. The prior group represents novel mathematical formulae for anomaly scoring, and the latter represents a general trend toward removing hyperparameters from algorithms through generalization. These new techniques are then understood in the context of feature modelling and more general anomaly detection techniques in Section 4.1.

Chapter 2: Modifications to FRaC

2.1 Entropy Normalized FRaC

2.1.1 Motivation

In FRaC, each feature makes a contribution to the normalized surprisal outlier score. Some features tend to make larger contributions to the overall normalized surprisal than others, such as those that, given a predicted value, have many unlikely (high surprisal) true values. Each one is unlikely, but overall they may be likely, in which case predictions for the feature often generate high surprisal. In FRaC, we can see these situations reflected in the error model as a feature, where continuous distributions are wide continuous distribution and confusion matrices have many possibilities of low probability. One way of capturing this concept is with the entropy (or differential entropy in the continuous case) of a feature in the training set, though conditional information is lost in this calculation, because we are only looking at the input distribution, and not taking predictor accuracy into account.

The normalized surprisal score roughly tends toward 0 (though the actual expected value is dependent on the distributions of each predictor, as discussed in Section 1.3.4), as entropy (expected surprisal) values are subtracted from surprisal scores, however, perhaps counterintuitively, the subtraction of entropy values actually has no effect on the overall ordering of surprisal values, and high entropy features often make much larger contributions to normalized surprisal scores. We can see this more easily if we rearrange the normalized surprisal equation:

$$
\begin{aligned}
&\mathrm{NS}(\hat{\mathbf{x}}) = \\
&\sum_{i=1}^{F}\sum_{m=1}^{M}\left\{\begin{array}{l} 0 \;\text{ if } \hat{\mathbf{x}}_i \text{ is missing, else:} \\ \mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i \mid C_{i,m}(\rho_i(\hat{\mathbf{x}})))\right) - \mathrm{H}(\hat{\mathbf{v}}_i^T) \end{array}\right. \\
&= \sum_{i=1}^{F}\sum_{m=1}^{M}\left\{\begin{array}{l} \mathrm{H}(\hat{\mathbf{v}}_i^T) - \mathrm{H}(\hat{\mathbf{v}}_i^T) \;\text{ if } \hat{\mathbf{x}}_i \text{ missing:} \\ \mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i \mid C_{i,m}(\rho_i(\hat{\mathbf{x}})))\right) - \mathrm{H}(\hat{\mathbf{v}}_i^T) \end{array}\right. \\
&= \sum_{i=1}^{F}\sum_{m=1}^{M}\left\{\begin{array}{l} \mathrm{H}(\hat{\mathbf{v}}_i^T) \;\text{ if } \hat{\mathbf{x}}_i \text{ is missing, else:} \\ \mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i \mid C_{i,m}(\rho_i(\hat{\mathbf{x}})))\right) \end{array}\right. \\
&\qquad - \sum_{i=1}^{F}\sum_{m=1}^{M}\mathrm{H}(\hat{\mathbf{v}}_i^T) \\
&= \sum_{i=1}^{F}\sum_{m=1}^{M}\left\{\begin{array}{l} \mathrm{H}(\hat{\mathbf{v}}_i^T) \;\text{ if } \hat{\mathbf{x}}_i \text{ is missing, else:} \\ \mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i \mid C_{i,m}(\rho_i(\hat{\mathbf{x}})))\right) \end{array}\right. \\
&\qquad - \sum_{i=1}^{F} M \cdot \mathrm{H}(\hat{\mathbf{v}}_i^T)
\end{aligned}
$$

As one can see, entropy is only visible on the left for missing values , and as a constant that is subtracted from each sample's normalized surprisal score. The constant has no effect on outlier rankings, because it is subtracted from *each sample* in a test set.

We have now seen that, when compared to average surprisal, normalized surprisal does not have an effect on outlier rankings in the absence of missing data. This means that normalized surprisal *does not* account for differences in the strength of the contributions of the surprisal values generated by different features in the final score. In order to do this, we would need to *divide* by the *expected surprisal.* I discuss a formal definition for *entropy normalized FRaC* (eFRaC) in the next section.

2.1.2 Algorithm Definition

In this section, I present the eFRaC algorithm, and the associated outlier score *entropy normalized surprisal*. I first present an intuitive naïve definition, and then after discussing issues with the naïve definition, proceed to define a final version that does not suffer the same inadequacies.

Naïve Definition

I define the *naïve entropy divided normalized surprisal* as follows:

$$\text{Naïve}\,\text{ENS}(\hat{\mathbf{x}}) = \sum_{i=1}^{F}\sum_{m=1}^{M}\frac{\left\{\begin{array}{l} 0 \;\;\text{if } \hat{\mathbf{x}}_i \text{ is missing, else:} \\ \mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i \mid C_{i,m}(\rho_i(\hat{\mathbf{x}})))\right) - \mathrm{H}(\hat{\mathbf{v}}_i^T) \end{array}\right.}{\mathrm{H}(\hat{\mathbf{v}}_i^T)} \tag{2.1}$$

This definition performs very well over discrete features (see Section 2.1.4 for empirical data supporting this claim), but the technique breaks down over continuous features. It is also worth noting that the technique is not defined over features of zero entropy, as this results in indeterminate forms of the form $\frac{0}{0}$, however these situations can be precluded by applying Laplace smoothing to discrete feature distributions before taking their entropy.

The greater problem is the issue of negative differential entropy values, which can occur when a PDF attains values in excess of 1. When terms are divided by these negative differential entropy values, highly surprising events are multiplied by a negative number, becoming highly negative, and thus push the score of a sample towards normalcy. Similarly, unsurprising events push the score toward anomality. Effectively, features with negative entropy have the opposite of the desired effect, and because of them, naïve entropy divided normalized surprisal is a poor outlier score.

Full Definition

There are several ways to modify the naïve eFRaC definition to overcome the shortcomings on continuous variables, but the simplest is to scale each continuous feature such that their distribution has a maximal value of one. Upon application of this scaling function, no value has negative surprisal, and consequently all features have nonnegative entropy.

Aside from the feature scaling step, the full eFRaC algorithm is the same as its naïve counterpart, so the formula is identical to Equation 2.1. The feature scaling can be implemented as a preprocessing step where the distribution of each feature over the training set is calculated, and subsequently, the training and test sets are scaled accordingly.

In many respects, this is similar to z-score normalization. With z-score normalization, if the data are assumed to be Gaussian distributed, the probability distribution function of a normalized feature has a maximum value of $f(x) \approx .4$ at $x = 0$, so the problem of negative surprisal values vanishes. Additionally, both techniques address the problem of surprisal over continuous distributions not being invariant under transformations by providing a technique to standardize a probability distribution.

2.1.3 Analysis of eFRaC

It is tempting to think of eFRaC as a sort of normalized FRaC, where a sort of normalization is being applied to the surprisal values of each feature. In many ways, this is not a bad analogy, as eFRaC is intended to smooth the contribution each feature makes to the overall anomaly score.

However, the analogy between entropy normalized surprisal and a featurewise normalization[1] of surprisal values only goes so far: with featurewise normalization techniques, *all* features are weighted equally, *including* features with little or no utility in classifying anomalies (such as those with near-uniform error model distributions). In eFRaC, only the *entropy* of the *true distribution* is used in performing the normalization-like step, so difficult to predict features still make a small contribution to overall anomaly scores, even if they have high-entropy true distributions.

For this reason, the experiment in (Noto et al., 2012) wherein FRaC is shown to be resistant to large numbers of irrelevant (noise) features could be repeated with eFRaC.

Similar to the normalization of continuous features prior to the use of distance based algorithms, such as KNN (Cover and Hart, 1967), in which the magnitude of each continuous feature is ignored, eFRaC allows FRaC to treat features equally with regards to their entropy, which in

[1] By featurewise normalization, I refer to any technique where the featurewise surprisal scores are normalized across the query set before being summed.

some sense is a proxy for the difficulty of a classification problem.

In addition to analyzing the anomaly score used by eFRaC, we need also concern ourselves with the effects of transforming the continuous features. As shown in Section 1.3.4, FRaC is scaling invariant under certain certain assumptions, so in these cases, the transformation has no side effects. When these assumptions are not met, it's not clear that the transformations will have negative side effects: if the features are normally distributed, the effects are very similar to z-score normalization, differing only by a global scale. In other cases, the results may be unpredictable.

If this transformation is thought to have unacceptable side effects, it may be reversed after calculating entropy values, in which case it has no effect on learners. These entropy values can then be scored, and used when calculating entropy normalized normalized surprisal.

2.1.4 Empirical Evaluation

In Table 2.1, I present an empirical comparison of FRaC, naïve eFRaC and eFRaC. The datasets presented in the table are from the UCI machine learning repository (Lichman, 2013). As in (Noto et al., 2012), these datasets are drawn from classification problems in the UCI machine learning repository (Lichman, 2013). They are converted to anomaly detection problems by denoting the dominant class *normal*, and all other classes *anomalous*. Fewer datasets are shown here than in (Noto et al., 2012) because the implementation of FRaC used here does not tolerate unknown values in the training set, so sets containing unknown values in the training set were removed.

In the table, AUROC values are reported by algorithm for each dataset. Because AUROC values are often so similar, sometimes the *mean rank* (calculated using 1224 ranking) is more informative than the raw mean. The number of times an algorithm scores maximally are all reported. When these values agree, a result may be considered to be more significant than any one of them alone.

Additionally, p-values from a two-tailed paired t-test are presented, both between FRaC-eFRaC and FRaC-neFRaC. Pairings are between the runs of each algorithm on a dataset. Whether error values between pairs are normally distributed is questionable, so these p-values may be overly optimistic.

In each experiment, a decision tree and a linear SVM were used to predict discrete and continuous features, respectively. Additionally, rather than running with leave one out cross validation (LOOCV), 16 fold cross validation was used instead. For these reasons, performance of FRaC in these experiments is lower than that reported in (Noto et al., 2010, 2012), but unless otherwise noted, this parameterization is consistently used, so the comparison is fair.

In the table, we see that naïve eFRaC performs worse than ordinary FRaC by a statistically significant amount, and ordinary eFRaC has performance similar to that of FRaC. However, eFRaC is more often maximal than FRaC, and if the abysmal performance of eFRaC over the *dermatology* dataset (AUROC of 0.0176 against FRaC's 0.9938) is removed from the dataset, the paired t-test has a p-value of 0.495. Presumably, reweighting by inverse entropy creates an extremely unlikely antagonistic weighting for this dataset, causing the poor performance of eFRaC. Since eFRaC is essentially a heuristic to obtain a better weighting for each feature, the existence of such antagonistic datasets is not surprising.

All in all, entropy normalized FRaC is a technique to control the weight of the contribution each feature makes to the overall outlier score, intended to make each feature equipotent. Empirical evidence suggests that this often leads to improvements, but there are times when the weightings generated by entropy normalized FRaC are inferior to the default uniform weighting of traditional FRaC.

Table 2.1: Comparison of FRaC, naïve eFRaC (neFRaC), and eFRaC over UCI Datasets

	FRaC	neFRaC	eFRaC	Real Features	Nominal Features	Training Set Size	Query Set Size
abalone	0.4083	**0.5555**	0.4168	**8**	0	1146	3031
acute	0.9523	**0.9756**	0.9629	1	**6**	45	75
audiology	0.7292	0.8635	**0.8651**	0	**52**	36	164
balance-scale	**0.95**	0.9137	0.9127	**4**	0	216	409
car	0.7872	0.766	**0.8091**	0	**6**	907	821
cmc	0.3905	**0.4317**	0.4034	2	**7**	471	1002
connect-4	**0.6431**	0.5681	0.5682	0	**42**	33354	34203
dermatology	**0.9938**	0.018	0.0176	2	**32**	84	282
ecoli	**0.8558**	0.1892	0.8298	**5**	2	107	229
glass	0.6678	0.439	**0.6892**	**9**	0	57	157
haberman	0.7035	**0.7186**	0.7005	**3**	0	168	138
hayes-roth	0.8514	**0.8533**	0.8029	0	**4**	38	94
image	0.9549	0.8264	**1**	**19**	0	22	188
ionosphere	**0.9673**	0.9135	0.967	**34**	0	168	183
iris	**1**	0	**1**	**4**	0	37	113
letter-recognition	0.993	0.9765	**0.995**	**16**	0	609	19391
magic	0.7325	0.4784	**0.8134**	**10**	0	9249	9771
nursery	**1**	**1**	**1**	0	**8**	3240	9720
page-blocks	0.591	0.3347	**0.9454**	**10**	0	3684	1789
parkinsons	0.6526	0.232	**0.6909**	**22**	0	110	85
pima-indians-diabetes	0.6564	0.4881	**0.6916**	**8**	0	375	393
poker	0.506	0.4991	**0.5341**	**5**	**5**	9369	15641
spambase	0.5258	0.5804	**0.8236**	**57**	0	2091	2510
tae	0.5117	0.3959	**0.5218**	1	**4**	39	112
tic-tac-toe	0.8322	0.8396	**0.8449**	0	**9**	469	489
voting-records	0.9182	0.9553	**0.9585**	0	**16**	200	235
wine	0.9273	0.8925	**0.9393**	**13**	0	53	125
yeast	0.693	0.3063	**0.7046**	**8**	0	347	1137
zoo	**1**	**1**	**1**	1	**15**	30	71
Mean	0.7722	0.6211	**0.7727**	8.3448	7.1724	2301	3536
Mean Rank	1.9655	2.3103	**1.4828**				
Maximal Count	8	7	**19**				
p-value		**0.0091**	0.9903				

Data from 29 UCI datasets shown. Color used to direct attention to more accurate algorithms, real or nominal feature dominated datasets, small training or test set sizes and more significant p-values.

2.2 Combinatorial Ensemble FRaC

Ensemblification in the typical machine learning sense is reviewed in Section 1.2.3. In this section, I discuss ensemblification in a second sense that is intrinsically tied to the FRaC algorithm, and introduce an algorithm that uses the information produced by an ensemble more effectively.

In FRaC, there is a second option for ensemblification. Each feature i has M classifiers associated with it, and as can be seen in the normalized surprisal equation (Equation 1.2), the surprisal values of these classifiers are summed. I subsequently refer to a traditional ensemble (that is an ensemble of predictors that is treated as a single predictor by FRaC) as an *inner ensemble*, and the FRaC-specific ensemble as an *outer ensemble.*

Inner ensembles have the advantage of being quite accurate, and therefore generally producing higher surprisal values on misclassifications, *however*, while this may be true over large numbers of samples, over individual samples, the idiosyncrasies and biases of various classifiers, which are "smoothed out" in an inner ensemble, sometimes produce high surprisal values that are very useful in detecting outliers. See see Figure 2.1 for a diagram showcasing their structural differences in the context of the FRaC algorithm, and see Figure 2.2 for an example comparing the surprisal scores produced by inner and outer ensembles.

The performance of FRaC using both an inner and an outer ensemble is shown in Table 2.2. Overall, their performance is quite similar although, perhaps surprisingly, the simpler inner ensemble seems to win out by a small margin. Though this result is somewhat consistent, it is not unlikely that the difference in the performance of inner and outer FRaC is entirely due to chance.

2.2.1 Analysis of FRaC Ensemble Use

In FRaC, the output of each learner in an outer ensemble is used to calculate surprisal values, which are simply summed together. This works reasonably well, as the information from each learner can be used to detect different types of surprising events. However, the one to one correspondence between error models and classifiers, paired with the simple summation, means that the results of each classifier can only be used independently.

Note that a sum of surprisal values is equal to the surprisal of the product of the probabilities that generated the surprisal values. If we make the hypothesis that the probability of the true value given each predicted value is independent from the other predictions, then what we are calculating with this sum is the surprisal of the probability of making *all* of the observations. Formally stated, under the independence hypothesis (and ignoring for the moment the possibility of missing values):

$$\mathrm{FS}(\vec{x}, i) = \sum_{m=1}^{M} \mathrm{I}\left(\mathbb{P}(x_{qi} \mid C_{p,i}(\rho_i(\vec{x_q})), A_{p,i})\right)$$

$$= \mathrm{I}\left(\mathbb{P}\left(\vec{x}_i \middle| C_{i,1}(\rho_i(\vec{x}))\right) * \ldots * \mathbb{P}\left(x_i \middle| C_{i,M}(\rho_i(\vec{x}))\right)\right)$$

$$= \mathrm{I}\left(\mathbb{P}\left(\vec{x}_i \middle| C_{i,1}(\rho_i(\vec{x})) \cap \ldots \cap \vec{x}_i \middle| C_{i,M}(\rho_i(\vec{x}))\right)\right)$$

It is highly unlikely for the surprisal values of multiple predictors to be independent, because the goal of each is to determine the same quantity, the surprisal of a particular feature given the remaining features and some training instances, using a particular training algorithm and parameterization. If we then assume that the predictions of each learner are *not* independent, we potentially have more information with which to calculate surprisal values.

2.2.2 Algorithm Definition

With the motivation of Combinatorial Ensemble FRaC (cFRaC) introduced, in this section I outline the algorithm itself.

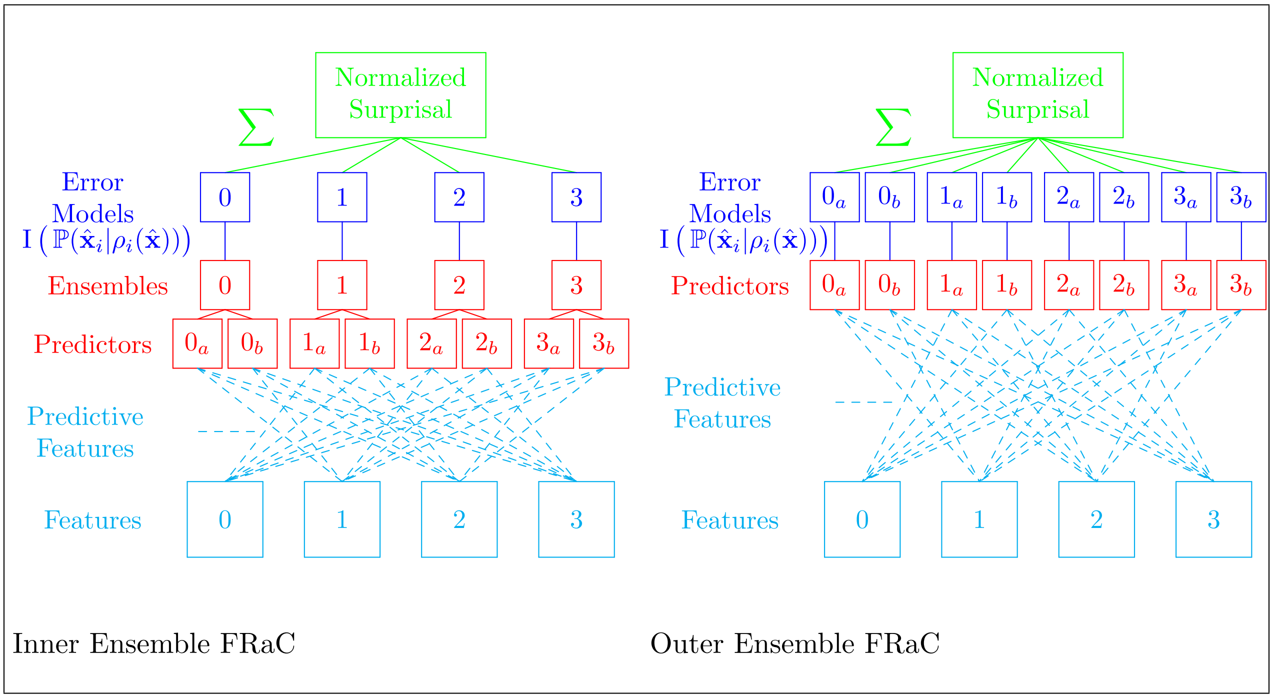


Figure 2.1: Visual Comparison of Inner and Outer Ensemble FRaC

Classifier **A**:

$$\begin{array}{cc} & \text{Predicted} \\ \text{True} & \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \end{array}$$

Classifier **B**:

$$\begin{array}{cc} & \text{Predicted} \\ \text{True} & \begin{bmatrix} .9 & .3 \\ .1 & .7 \end{bmatrix} \end{array}$$

Classifier **C**:

$$\begin{array}{cc} & \text{Predicted} \\ \text{True} & \begin{bmatrix} .7 & .3 \\ .3 & .7 \end{bmatrix} \end{array}$$

Ensemble of **A**, **B**, **C**:

$$\begin{array}{cc} & \text{Predicted} \\ \text{True} & \begin{bmatrix} .75 & .2 \\ .25 & .8 \end{bmatrix} \end{array}$$

Now, suppose that we used the inner ensemble to calculate the surprisal of a class 0 feature being predicted to be in class 1. We would get $-\log_2(.25) \approx 2$ bits. The outer ensemble would give (assuming all predictors made the same prediction) an average surprisal of:

$$-\frac{\log_2(.3) + \log_2(.1) + \log_2(.3)}{3} \approx 2.265 \text{ bits.}$$

However, let us now suppose that we have a class 1 feature being predicted to be in class 0. The inner ensemble surprisal is $-\log_2(.2) \approx 2.322$ bits, and the outer distribution surprisal is:

$$-\frac{\log_2(.2) + \log(.2) + log(.3)}{3} \approx 2.127 \text{ bits.}$$

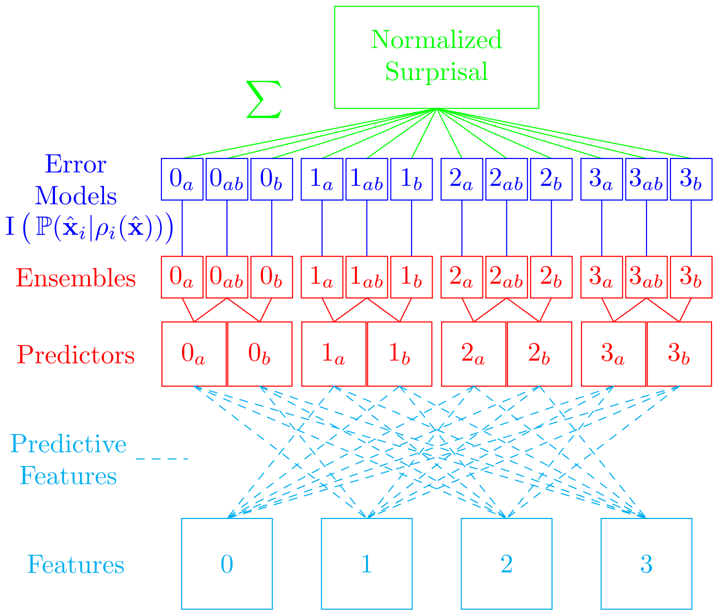
Let's summarize these results in a matrix.

(Average) surprisal values:

(Predicted, True)	Inner	Outer
(1, 0)	2	2.265
(0, 1)	2.322	2.127

In this example, we see that sometimes an inner ensemble has a higher surprisal contribution than the outer ensemble, and sometimes it does not. In Table 2.2, we empirically verify that neither inner nor outer ensembles are consistently superior for the anomaly detection problem using FRaC.

Figure 2.2: Inner vs. Outer Ensembles in FRaC


Normalized Surprisal
$\sum$
Error Models
$\mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i|\rho_i(\hat{\mathbf{x}}))\right)$
0_a 0_{ab} 0_b 1_a 1_{ab} 1_b 2_a 2_{ab} 2_b 3_a 3_{ab} 3_b
Ensembles
0_a 0_{ab} 0_b 1_a 1_{ab} 1_b 2_a 2_{ab} 2_b 3_a 3_{ab} 3_b
Predictors
0_a 0_b 1_a 1_b 2_a 2_b 3_a 3_b
Predictive Features
Features
0
1
2
3

In this diagram we see a visual presentation of the cFRaC algorithm. In this instance, models are constructed with two predictors over four features, resulting in a total of $2^2 - 1 = 3$ ensembles and error models.
Note that the set of error models and ensembles is a superset of both those that would be present in inner ensemble and outer ensemble FRaC. Due to the difficulty of fitting an exponential number of ensembles onto the page, only two predictors for each feature are shown, however for larger predictor quantities, feature combinations that are not represented in inner or outer ensemble FRaC appear.

Figure 2.3: Visual Diagram of the cFRaC Algorithm

In FRaC, we have a one to one correspondence between prediction models and error models. All surprisal calculations take the form $\mathrm{I}(\mathbb{P}(\mathrm{T}\,|\,\text{prediction } i)$, where T represents the true value, and these values are summed for each prediction i. Ultimately, this is the naïve version (so called because of the independence assumption) of $\mathrm{I}(\mathbb{P}(\mathrm{T}\mid\vec{P}))$, where $\vec{P}$ is a vector of all predictions for this feature, and we would get more information if we were to use this second form. However, the evidence space grows exponentially with the number of predictors, making it extremely difficult to obtain enough data to build an error model[2].

There are several options available to solve this problem: in the discrete case, we can "bleed" classification events with a multidimensional convolution matrix, not unlike the Gaussian blur technique in image processing (e.g. if events A and B differ only by one predictor, their probabilities should bleed into each other because they are very similar). This essentially artificially populates the matrix with likely events, and in the continuous cases, we can perform similar operations with Gaussian mixture distributions. However, there is an alternative and much simpler way to build an error model with a more complicated evidence space: ensemblification.

An inner ensemble's error model looks just like an ordinary ensemble, as an inner ensemble is just an ordinary prediction model. However, we can consider the ensemble's prediction to be evidence, and after each predictor makes its prediction, we combine the predictions in all possible ways (except for combination into the null set: this combination of predictors yields no information) to form new models, and build an error model for each one. Now, given N predictors, we train $2^N - 1$ error models, and for each training instance, for each combination of predictors, we ensemblify the predictions. We then use these ensemblified predictions to build and query error models. See Figure 2.3 for a visual diagram outlining the algorithm.

This technique results in the combination of

[2] This is highly similar to the problem of training a Bayesian classifier without the naïve independence assumption.

a large number of error models, each of which is queried using some subset of the information carried by the results of all predictors. Because the ensemble of all predictors is included in the set of all combinations, the inner ensemble of all predictors is represented with an ensemble. The outer ensemble is also represented, as each singleton predictor is represented in the combinatoric set. Additionally, all intermediate combinations are also represented, which is potentially very useful, because it allows powerful combinations of predictors to self select (as they will have high surprisal events in their error models).

Because the contribution of both the inner and outer ensemble, as well as intermediate ensembles, are considered in combinatorial ensemble FRaC, it is not unreasonable to expect superior accuracy to either technique alone. The fact that FRaC inherently self selects through the construction of error models furthers that hypothesis, as the algorithm is able to weigh different subset ensembles differently based on their accuracy values. Empirical justification of these claims is provided in Section 2.2.3.

Formal Definition

Given N predictors, the feature surprisal for a known feature q for a feature vector $\vec{x}$ is given by

$$\mathrm{CNS}(\hat{\mathbf{x}}) = \sum_{i=1}^{F}\sum_{e=1}^{2^M}\begin{cases} 0 \;\text{ if } \hat{\mathbf{x}}_i \text{ is missing, else:} \\ \mathrm{I}\Big(\mathbb{P}\big(\hat{\mathbf{x}}_i \mid E_{i,e}(\rho_i(\hat{\mathbf{x}}))\big)\Big) - \mathrm{H}(\hat{\mathbf{v}}_i^T) \end{cases} \tag{2.2}$$

where $E_{i,e}$, for some i, represents one possible ensemble of the N predictors of the i^{th} feature, such that no two $e \in [1, 2^M]$ are the same. Note that an ensemble of classifiers may produce a tie. There are many ways to resolve this situation, but for simplicity ties are broken randomly.

2.2.3 Empirical Evaluation

In this section, I provide empirical evidence to suggest that Ensemble FRaC is superior to inner and outer ensemble FRaC. In Table 2.2, we can see that the average AUROC of cFRaC is higher than that of both inner and outer ensemble FRaC, and similarly the average rank is lower. Inner ensemble FRaC has more maximal rank datasets, this may be because it is the simplest of the three algorithms and it does very well on simple datasets. Overall, the performance of inner and outer ensemble FRaC are similar, and combinatorial ensemble FRaC is slightly better.

Each of these experiments involves an ensemble, so in addition to the decision tree and linear SVM used in previous experiments, a 3 layer neural network classifier and a decision tree regressor were added. Note that when the ensembles are of size 2, the normalized surprisal score of a sample in cFRaC is equal to the sum of normalized surprisal scores in the inner and outer ensemble FRaC algorithms. For this reason, we can think of combinatorial ensemble FRaC for an ensemble of size 2 as an ensemble anomaly detection technique, where the component anomaly detectors are inner and outer ensemble FRaC. We see from the Table 2.2 that they make diverse predictions, and both are accurate in that they have mean AUROC values in excess of 0.5, so from ensemble theory it follows that cFRaC should be superior to either one. For this reason, in addition to the 2-tail t-tests, single tail tests are provided as well, to test for superior performance of cFRaC over inner and outer ensemble FRaC.

From the t-tests, we see that the experiment shows no statistically significant difference between inner and outer ensemble FRaC. There may be a small difference in performance, but vastly more powerful experiments would be needed to find it. We also see that neither of the two tailed tests for cFRaC is significant, however the one tailed test between cFRaC and outer ensemble FRaC is significant, and similarly the test between cFRaC and inner ensemble FRaC is suggestive.

	Inner Ensemble	Outer Ensemble	Combinatorial FraC	Real Features	Nominal Features	Training Set Size	Query Set Size
abalone	**0.4191**	0.4158	0.4159	**8**	0	1146	3031
acute	**1**	0.9799	**1**	1	**6**	45	75
audiology	0.7429	0.7467	**0.75**	0	**52**	36	164
balance-scale	**0.9428**	0.9191	0.936	**4**	0	216	409
car	0.7925	**0.8789**	0.8738	0	**6**	907	821
cmc	**0.4039**	0.3937	0.396	2	**7**	471	1002
connect-4	0.5851	0.6047	**0.6078**	0	**42**	33354	34203
dermatology	**0.9941**	0.9934	**0.9941**	2	**32**	84	282
ecoli	**0.9636**	0.9491	0.9596	**5**	2	107	229
glass	**0.7277**	0.7021	0.7151	**9**	0	57	157
haberman	0.6582	**0.6662**	0.6628	**3**	0	168	138
hayes-roth	**0.8875**	0.8827	**0.8875**	0	**4**	38	94
image	0.9993	**1**	**1**	**19**	0	22	188
ionosphere	0.9663	**0.9694**	0.9691	**34**	0	168	183
iris	**1**	**1**	**1**	**4**	0	37	113
letter-recognition	**0.9985**	0.9854	0.9919	**16**	0	609	19391
magic	0.8224	**0.8538**	0.8494	**10**	0	9249	9771
nursery	**1**	**1**	**1**	0	**8**	3240	9720
page-blocks	**0.8676**	0.7747	0.7988	**10**	0	3684	1789
parkinsons	0.5805	**0.5901**	0.5805	**22**	0	110	85
pima-indians-diabetes	**0.684**	0.6327	0.6599	**8**	0	375	393
poker	**0.5557**	0.5551	0.5539	**5**	**5**	9369	15641
spambase	0.7728	0.7809	**0.7826**	**57**	0	2091	2510
tae	0.4953	**0.5653**	0.5427	1	**4**	39	112
tic-tac-toe	0.9043	**0.9951**	0.9945	0	**9**	469	489
voting-records	0.915	0.9101	**0.9161**	0	**16**	200	235
wine	0.893	0.9045	**0.905**	**13**	0	53	125
yeast	**0.7189**	0.6884	0.7032	**8**	0	347	1137
zoo	**1**	**1**	**1**	1	**15**	30	71
Mean	0.8031	0.8048	**0.8085**	8.3448	7.1724	2301	3536
Mean Rank	1.8276	2.069	**1.6207**				
Maximal Count	**16**	11	12,				
p-value (2 tail paired)	inner, outer	0.8122	0.3292, 0.0569				
p-value (1 tail paired)			0.1646, **0.0284**				

Table 2.2: Comparison of inner and outer ensembles vs. combinatorial FRaC over UCI datasets. p-values shown between inner and outer ensemble FRaC, and also between cFRaC and both of the previous techniques. cFRaC-inner ensemble and cFRaC-outer ensemble two tailed p-values are shown, comma separated, in that order, as are their single tailed variants.

2.3 Missing Value Corrected FRaC

2.3.1 Motivation

In FRaC, normalized surprisal (See Equation 1.2) is used to calculate an outlier score. This metric has a slight bias toward 0 for samples containing missing features, because missing values contribute nothing to normalized surprisal scores. If we carry this to the extreme, a sample consisting of only missing values would have a normalized surprisal score of 0.

Now, let us suppose we have a sample s of F features, with an anomaly score of x. Now we remove a random feature from s and replace it with the value unknown. Let us denote this modified sample s'. The effect of this operation on the surprisal of the remaining feature values is difficult to predict; it depends on the classifier, and the technique used to handle unknown values, as well as the feature values themselves. If we suppose that it has a net effect of zero on the surprisal values of the other features, then the expected normalized surprisal of s' is $x \cdot \frac{F-1}{F}$. As we continue removing features, this value keeps dropping, until eventually, when all features are missing the normalized surprisal is 0.

In the next section, I introduce a modified algorithm, missing value corrected FRaC, which corrects for this effect.

2.3.2 Algorithm Definition

We can correct for this drop in absolute surprisal suggested in the previous section by multiplying the anomaly score x by $\frac{F}{F-1}$, which we can generalize to $\frac{F}{F-m}$ for m missing features. This yields a new formula:

$$\mathrm{NS_m}(\hat{\mathbf{x}}) = \sum_{i=1}^{F} \sum_{m=1}^{M_i} \left\{ \begin{array}{l} 0 \;\text{ if } \hat{\mathbf{x}}_i \text{ is missing, else:} \\ \mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i \mid C_{i,m}(\rho_i(\hat{\mathbf{x}})))\right) - \mathrm{H}(\hat{\mathbf{v}}_i^T) \end{array} \right. \cdot \frac{F}{\sum_{i=1}^{F} \left\{ \begin{array}{l} 0 \text{ if } \hat{\mathbf{x}}_i \text{ is missing} \\ 1 \text{ otherwise} \end{array} \right.} \tag{2.3}$$

Using $\mathrm{NS_m}$ (Normalized Surprisal Missing value corrected) in place of ordinary normalized surprisal yields the alternative algorithm, mFRaC (missing value corrected FRaC). Note that in mFRaC, a sample that consists entirely of missing values has an undefined anomaly score. Because such a sample contains no information (assuming that the absence of a feature does not carry information), refusing to assign an outlier score in such a case is not unreasonable.

2.3.3 Analysis

In the previous derivation, we made the somewhat dubious assumption that replacing a feature with an unknown had no effect on the surprisal values of the remaining features. If we do not make this assumption, we need to begin looking at actual data to see how well the assumption holds.

Logically, the more effect a feature has on predicting the remainder of the features (classifier and dataset dependent), the less the assumption holds. In data with more features, individual features are less important to predicting the other features, so the assumption should hold up better on such datasets. Additionally, the assumption may be less inaccurate when small numbers of features are missing, as the remaining features are more likely to compensate for the missing features[3].

It is important to consider the types of datasets in which we can expect differences in the output of FRaC and mFRaC. When no features are missing, the corrective term in mFRaC has value 1, so the two algorithms produce identical outlier scores on the same data.

Similarly, when every element of the test set has the same number missing values, the same multiplicative factor is applied to each sample, so the ranking of the anomaly scores of the test set does not change.

[3] This statement is highly dependent on the dataset in question. In nonredundant datasets, such as those where features sum or exclusive or to some value, we can not use the remaining features to compensate for a missing feature.

It is only in cases where different elements of the test set of different numbers of missing values that mFRaC has an effect on the anomaly ranking. Essentially, in these cases, mFRaC allows a sample with more missing features to be more fairly compared to a sample with fewer missing features, by compensating for the fact that fewer surprisal values are summed in calculating the normalized surprisal of the sample with fewer known features.

2.3.4 Empirical Evaluation

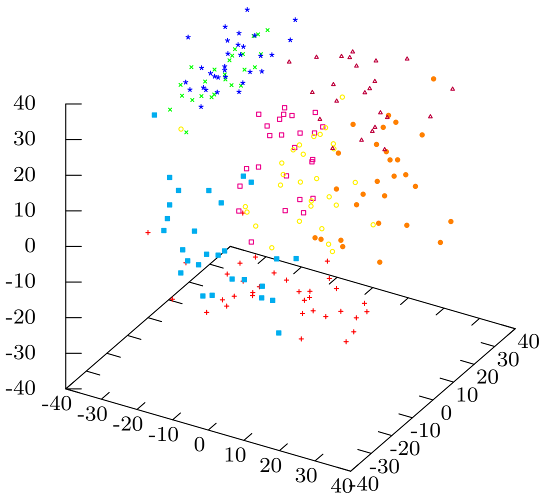
Experimental validation on synthetic data is provided in Figure 2.5. In this experiment, we see that mFRaC slightly outperforms traditional FRaC, especially when the range of missing value counts is large.

Additionally, a comparison of mFRaC and FRaC over UCI datasets is provided in Table 2.3. In addition to testing these algorithms against the vanilla UCI dataset, the same experiment was repeated using modified datasets where 25% of the data points were replaced with unknown values. Most of the UCI repository datasets don't have many missing values, so this was done to test the performance of mFRaC in the sort of data-starved scenario in which it was designed to be used.

It is difficult to extract a conclusion from these data over the raw UCI datasets; the mean AUROC of mFRaC is higher than that of FRaC, but FRaC produces more maximal AUROC values and has a slightly lower average rank (though when comparing only two predictors the last two imply one another). Additionally, on the artificially sparsified data, mFRaC's performance seems to be superior to that of ordinary FRaC. However, the differences are quite small, and do not appear to be statistically significant.

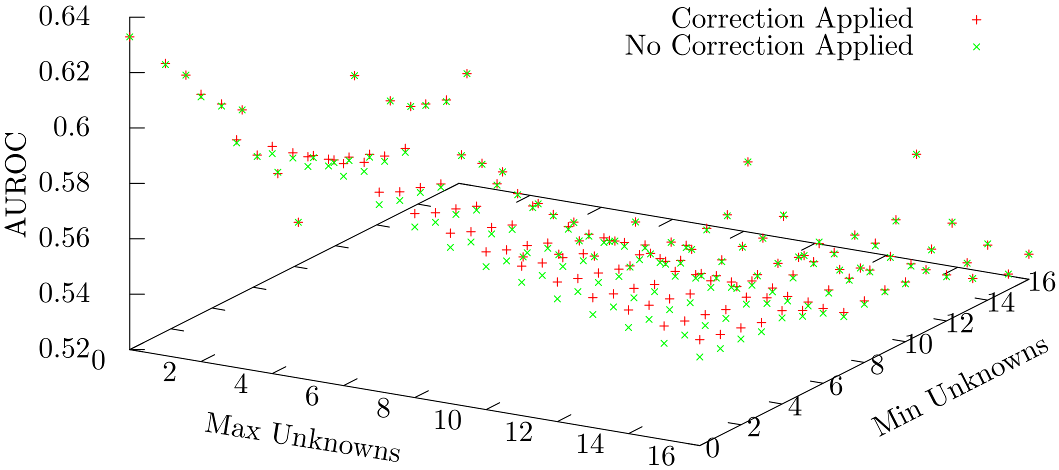
	FRaC	mFRaC	FraC 0.25	mFRaC 0.25
abalone	**0.4083**	0.4024	**0.5301**	0.5252
acute	0.9523	**0.9544**	0.8734	**0.9354**
audiology	**0.7292**	0.7248	0.6891	**0.7379**
balance-scale	**0.95**	0.9139	0.8119	**0.8133**
car	**0.7872**	0.7512	0.6003	**0.6249**
cmc	0.3905	**0.3982**	**0.4186**	0.3997
connect-4	0.6431	**0.6435**	**0.5542**	0.5514
dermatology	**0.9938**	0.9931	0.8448	**0.8538**
ecoli	**0.8558**	0.8489	0.8089	**0.8137**
glass	**0.6678**	**0.6678**	0.4777	**0.5734**
haberman	0.7035	**0.7135**	**0.6673**	0.6111
hayes-roth	**0.8514**	0.8495	0.6804	**0.7626**
image	0.9549	**0.9674**	**0.8118**	0.7674
ionosphere	0.9673	**0.969**	**0.8718**	0.8535
iris	**1**	**1**	**0.9942**	0.9938
letter-recognition	**0.993**	0.9818	**0.9333**	0.9195
magic	**0.7325**	0.6671	0.6408	**0.6508**
nursery	**1**	0.9355	0.8489	**0.8532**
page-blocks	0.591	**0.8432**	0.552	**0.5749**
parkinsons	0.6526	**0.7663**	**0.5954**	0.5099
pima-indians-diabetes	**0.6564**	0.623	0.5896	**0.5901**
poker	**0.506**	0.4548	**0.4952**	0.4857
spambase	**0.5258**	0.4665	**0.5674**	0.5613
tae	0.5117	**0.5389**	**0.5627**	0.4454
tic-tac-toe	0.8322	**0.8396**	0.6599	**0.6653**
voting-records	0.9182	**0.9202**	**0.9081**	0.8913
wine	**0.9273**	0.9091	0.8276	**0.8606**
yeast	0.693	**0.714**	**0.6048**	0.5287
zoo	**1**	0.9955	0.9455	**0.9606**
Mean	0.7722	**0.7742**	**0.7023**	0.7005
Mean Rank	**1.4138**	1.5172	1.5172	**1.4828**
Maximal Count	**17**	14	14	**15**
p-value		0.8553		0.8358

Table 2.3: Comparison of FRaC and mFRaC over UCI Datasets



Visualization of the first three dimensions of the first category of the test data. 8 sets of 4 dimensional Gaussian distributed points, centers chosen uniformly from a 4 dimensional hypercube, standard deviations chosen randomly from $[-16, 16]$. Each set is chosen from 8 possible distributions, and the selected distribution's index is added as a categorical variable, for a total of $8 * 4 = 32$ continuous variables and $8 * 1 = 8$ categorical variables.

Figure 2.4: Test Data For mFRaC-FRaC Comparisons.



FRaC and mFRaC were run on mixed data consisting of samples with varied numbers of missing data points. The x and y axes determine the minimum and maximum number of missing features for a trial, and the z coordinate is the AUROC score for said trial. Samples are drawn such that an equal number of each possible number of missing values is present.

Dataset is explained in Figure 2.4.

Figure 2.5: AUROC of mFRaC vs Traditional FRaC.

Chapter 3: Heuristic Improvements

3.1 Feature Selection Overview

3.1.1 Motivation

The FRaC algorithm does not scale well: training FRaC is $\Omega(f^2)$ for f the number of features. Here I present preprocessing techniques to perform feature selection for FRaC, with the goal of reducing the feature space to provide a reduced runtime.

Because FRaC cross validates each error model, and many of the algorithms used in classification and regression by FRaC perform implicit feature selection, feature selection algorithms cannot be expected to improve the *accuracy* of FRaC, however, by reducing the number of features, the *runtime* can be improved.

Because we are not expecting improved accuracy, these heuristic feature selection algorithms are evaluated not by their improvement in AUROC scores, but instead by a lack of reduction of AUROC score as compared to removing an equal number of random features. They are also evaluated by their computational cost, both in terms of asymptotic complexity and empirically validated time costs.

3.1.2 Feature Selection Techniques

Many traditional feature selection and feature reduction techniques operate on the principle of reducing redundancy in the feature space. This works very well for many machine learning applications, however the consequences of reducing redundancy in FRaC are severe: redundant or correlated features are used to predict each other, contributing potential high surprisal events when the redundancy or correlation is broken. When redundant or correlated features are removed, this signal is destroyed, and FRaC's performance is degraded. Principle Component Analysis (PCA, introduced in (Pearson, 1901)) is an example of a popular feature space reduction algorithm that succumbs to this flaw.

Additionally, wrapper method techniques involve *multiple* iterations of training, and are thus far slower than just training over all features[1]. FRaC can be considered to be an embedded method, as each feature's learner can choose to ignore every other feature (if the learner itself has embedded properties), and a feature's contributions to normalized surprisal can be neutralized by an error model that considers all events equally surprising. The remaining category of feature selection techniques, filter methods, are worth considering, so long as proxy criteria for FRaC accuracy can be found.

3.2 Filter Techniques

Filter methods are particularly relevant in high dimensional feature spaces. They have been used to great effect in computational biology on DNA microarrays (Xing et al., 2001) and in text classification (Guyon and Elisseeff, 2003): both of these domains can contain many thousands of features, and filtering is an efficient technique to reduce the space to tractable proportions.

These techniques provide scores to each feature, and the scores can be used to select a subset of features on which to perform FRaC. This selection can be done with an absolute cutoff, particularly when the filter score has some useful interpretation (remove all features with score less than x), or it can be done on a constant fraction (remove the bottom 25% of features by filter score). To be useful, filtering by cutoff generally requires a deep knowledge of the filter criterion, and sometimes also knowledge of the domain, so constant fraction filtering is the simpler alternative.

It is exceedingly difficult to concoct useful filter score algorithms for FRaC, because in FRaC, each feature interacts with every other feature, both in predicting and being predicted by them.

[1] If it were the case that training was *not* the bottleneck, as could happen with a sufficiently large query set, then wrapper methods could improve runtime. However, these techniques are targeted at improving training time, so these techniques are not discussed further.

Because of these complicated interactions, it is difficult to predict the effects of removing any particular feature.

Additionally, some traditional feature selection techniques designed for general purpose use with classification work in an antagonistic manner with FRaC. Most feature selection algorithms attempt to find and eliminate *irrelevant* features, which are not related to the pattern at hand, and *redundant features*, which are related, but have low information content in the context of the remaining features. Correlated features are generally redundant, and a feature like, for instance, eye color, is probably irrelevant to disease classification. In FRaC, removal of irrelevant features is helpful, however removal of redundant features is not: FRaC thrives on redundancy[2], and requires predictable relationships between features in order to function.

In order to truly find irrelevant features in FRaC, one must examine each feature in the context of every other feature to decide to what degree each feature is related to every other feature. This is as complex as the training step of FRaC, and therefore does not result in a reduction of runtime. For this reason, all of the filter score generation techniques below score features with heuristics designed to estimate the utility of a feature for use in FRaC. Because each has its advantages and disadvantages, and selects features for different reasons, it is both possible and recommended to use multiple scoring techniques in order to select the final set of features.

3.2.1 Partial Filtering

Ordinarily, filtering a feature removes it entirely from a dataset, however in FRaC an alternative exists. By default, FRaC trains an ensemble of classifiers for each feature, using every remaining feature. As an alternative, I introduce the FRaC-specific *partial filtering* technique. Instead of removing the features entirely, we leave them, allowing unfiltered features to train models based on them, but we *do not* train models for filtered features.

Partial filtering is a far more conservative metatechnique than the traditional filter method. Because partially filtered features remain available to learners for the unfiltered features, the decision to partially filter a feature is entirely local: no other feature is affected. For n the total number of features, any sort of analysis of the interdependence of features is $\Omega(n^2)$, so it is impossible to analyze the effects of removing a feature on predicting other features in subquadratic time. Partial filtering affords a solution linear in n by locally analyzing a feature, and knowing that the decision to remove it will not interfere with the prediction of other features.

3.2.2 Feature Scoring Algorithms

Correlation Approach

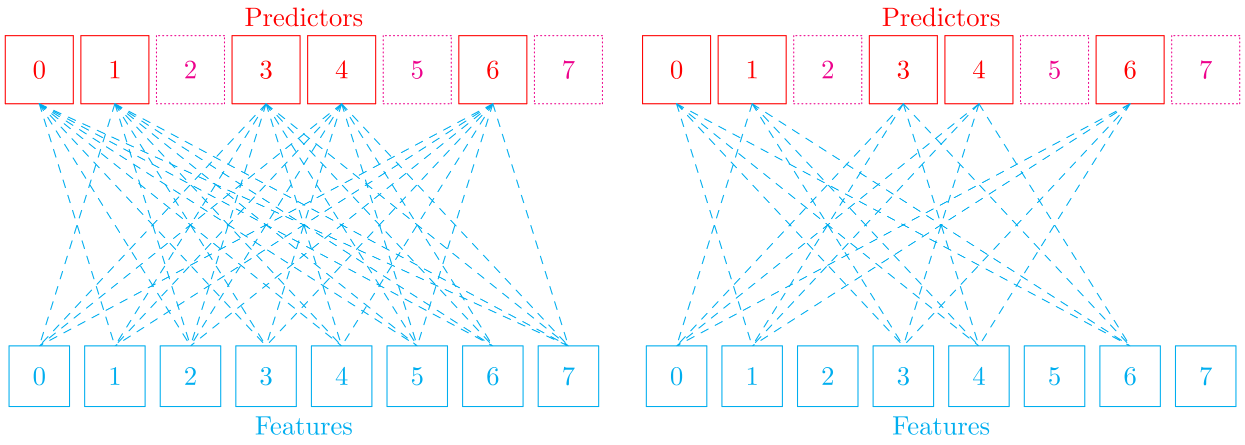
Though looking at correlation alone misses many of the subtler ways in which features can be related, and only applies to numeric features, determining which features are correlated is an excellent way to make sense of a dataset and begin finding relationships. Correlated features are redundant, so for many machine learning applications, they are undesirable, and a filter method could operate by identifying and removing them, as discussed in (Hall, 1999). In FRaC, this is not the case: redundant features are greatly beneficial, so long as the redundancy breaks down in anomalies. For this reason, features that correlate with other features, and are thus predictable, are highly desirable.

Both positive and negative correlations contribute to predictability, and this is reflected in the formula below.

$$\text{corrscore}(f) = \sum_{i=1}^{\dim F} \Big(|\text{corr}(F_i, F_f)| \Big) - 1$$

In this formula, F is a vector representing all

[2] I should qualify this statement: in FRaC, completely redundant features result in simple predictors, and if the redundancy is also present in anomalous data, FRaC will make correct predictions and fail to detect anomalies when subtler patterns (which would be picked up in the absence of the redundant feature) break down.



Partial Filtering of features 2, 5, and 7.

Full Filtering of features 2, 5, and 7.

Comparison of data flow between full and partial filtering. Note the richer connection network in the partial filtering diagram. No filtered feature has an error model, and thus no filtered feature contributes to the normalized surprisal score.

Figure 3.1: Full vs Partial Filtering Data Dependency Diagram

features, and corr is the function taking the Pearson correlation between two features over the training set. Alternatively, Spearman correlation may be used; the proper choice requires more sophisticated analysis or domain knowledge of the dataset. Notice that the correlation of a feature with itself is 1, and this is subtracted out, making this function take on values in the range $[0, \dim F - 1]$.

Although highly scoring features are definitely predictable, low scoring features may well be predictable as well. Unconditional correlation is a blunt instrument, but it is a necessarily blunt instrument, as more complex analysis of interfeature relationships takes more time. This technique is already $\theta(f^2 n)$ for f the number of features and n the number of training instances. Many filter criteria look at a feature in isolation, and have $\theta(fn)$ time complexities. This technique, while slower, is theoretically capable of teasing apart relationships in a way that is impossible in a linear algorithm.

KL Divergence

As outlined in Section 1.2.2, Kullback Leibler divergence is a metric used to evaluate the efficacy of using an encoding optimized for one probability distribution to send signals from another. KL divergence between classes, usually in the symmetric form, has been used as a filter criterion to great success in the classification field (Baker and McCallum, 1998; Schneider, 2004; Coetzee, 2005), as has Jensen-Shannon divergence (Coetzee, 2005).

Unlike the classification problem, in anomaly detection, ordinary asymmetric KL divergence has a logical interpretation: the predominant normal class can be considered the “true class”, and the divergence from the true class in the anomalous class can be measured and used as a filter criterion. Unfortunately, in the unsupervised anomaly detection problem, we do not have access to a collection of anomalous samples, so we can not construct feature distributions for the anomalous class.

Instead of looking at the KL divergence between features in normal and anomalous classes, we can look at KL divergence between the training and query set. For some feature, we consider the training set distribution to be the "true distribution", as it is our sampled approximation of the true distribution, following Laplace's reasoning and the continuous analogue outlined in 1.2.1. We then consider the query set to be the approximation distribution, because as a heterogeneous mixture of normal and anomalous samples, the features of the query set are potentially distributed differently from the training set. To identify these differences, for random variables X drawn from feature i of the training set, and Y drawn similarly from the query set, we take $\mathrm{D}_{\mathrm{KL}}(X \,\|\, Y)$ as a filter score.

Features with high KL divergence have significant differences in their distributions between the normal and anomalous classes, and thus, barring sampling artifacts, are valuable features to the FRaC algorithm. No conclusion can be drawn about features with low KL divergence, so in some sense this technique identifies which features *not* to remove, rather than those which should be removed. It is important to note that when anomalies are varied, or even when they are not, there may not be much difference between normal and anomalous feature distributions, even if the *relationships* between features have changed. For this reason, KL divergence based filtering can filter valuable features over less valuable features if they have similar low divergences. Low KL divergence filtering is thus a heuristic that can produce false positives as to what to filter, but it will rarely produce false negatives. It may also perform better on homogeneous anomaly types over heterogeneous anomaly types, because the former is more likely to have significant differences between feature distributions.

Technically, examining the query set makes this algorithm a *transduction* anomaly detection algorithm, and consequently changing the query set has the potential to change the anomaly score *of constant elements of the query set.* Transductive learning is easier than nontransductive, but transductivity precludes the use of an algorithm in an online setting.

I conclude the theoretical analysis of divergence based filter techniques with a brief argument regarding the ability of the divergence based techniques to remove irrelevant features. I also comment that the same holds for similar divergence-based approaches, including Jensen-Shannon filtering.

Proposition 2. *As the number of training and test instances tend to infinity, the filter scores of divergence based filtering techniques of irrelevant features tend to 0.*

Proof. Suppose that feature f is irrelevant to the classification problem at hand. Then the distribution of f in both the normal and anomalous classes is identical, as the negation would contradict the hypothesis. Now, as we draw s samples, the sampled approximation of a distribution d, denoted $f_d(s)$, approaches the true distribution, denoted f_{T}. The training set consists of only normal samples, and the query set is a mixed distribution of normal and abnormal samples, but regardless of the mixture ratio, the distribution of f over the query samples is identical to that over the training samples, because the distribution over normal and abnormal samples is identical. $\div XX = 0$, by positive definiteness of divergence functions, thus $\lim_{s\to\infty} \mathrm{D}_{\mathrm{KL}}(f_{\mathrm{norm}(s)} \,\|\, f_{\mathrm{anom}}(s)) = 0$, which we can interpret to mean that for irrelevant features, the KL divergence filter score goes to zero as the number of both training and query samples go to infinity. □

Entropy

Entropy is a valuable filter criterion insofar as it is a proxy for predictability. Low entropy features are highly predictable, and furthermore, rarer events in low entropy features may occur so rarely that insufficient information exists to build a statistically significant error model, thus making predictions for these rare events of little use. Because the predictions of common events will be low-surprisal, and predictions of rare events will not be statistically significant, no predictions from this feature are helpful.

$$\begin{array}{cc} & \text{Predicted Class} \\ \begin{array}{c}\text{True}\\\text{Class}\end{array} & \begin{bmatrix} 103 & 3 & 0 \\ 7 & 11 & 2 \\ 2 & 1 & 6 \end{bmatrix} \end{array} \qquad \begin{array}{cc} & \text{Predicted Class} \\ \begin{array}{c}\text{True}\\\text{Class}\end{array} & \begin{bmatrix} .904 & .222 & .091 \\ .070 & .667 & .273 \\ .026 & .111 & .636 \end{bmatrix} \end{array}$$

Key:
Very common, probability near 1.
Insignificant due to low column count.
Unlikely due to class imbalance bias.
Extremely insignificant due to low column count and an unlikely incorrect prediction.

Confusion matrices of counts and of column normalized Laplace smoothed probabilities.
Events in *gray* are unlikely to have a statistically significant number of entries, and are thus dominated by chance and Laplace Smoothing. Meanwhile, the black cell (correct prediction of the preeminent class is by far the most likely, and is a low surprisal event, and therefore makes a minimal contribution to overall surprisal. Therefore, regardless of the true value and prediction, the surprisal contribution for the feature is either minimal or subject to extreme sampling bias, and therefore not useful to FRaC.

Figure 3.2: Confusion Matrix of a Low Entropy Feature.

We can see this most clearly by analyzing the different cells of a confusion matrix. See Figure 3.2 for an example of this, as well as an in depth analysis of this of how different biases affect these types of confusion matrix. Note that the same principles apply to continuous variables as well.

The use of entropy as a filter criterion suffers from the same flaw as do all filter method techniques for semisupervised anomaly detection: without observing the difference between a feature in the normal class and an anomalous feature, we can not make a fully informed decision about which features to filter out. As a pathological example, consider a dataset in which a particular categorical feature is always A in the normal class, but never A on anomalies. The entropy of this class is of course 0 bits over the training set, so it would be filtered, even though it alone is a perfect anomaly detector.

One way to alleviate this problem is via transduction, where we simply take the entropy of the union of the training set and test set, and therefore we target the removal of features that are highly consistent even over anomalous samples.

It is important to note that entropy is not invariant to scaling, but scale should not be a factor in considering when to filter a feature. The technique described in Section 2.1.2, whereby all continuous variables are scaled such that all maximal values of their PDF are 1, allows us to fairly compare entropy values invariant of their original scale. This technique is not necessary for KL divergence, because KL divergence is scaling invariant, as discussed in Section 1.2.2.

High entropy features are also often not helpful in the anomaly detection problem, because high entropy features are generally more difficult to predict. Predictors for difficult to predict features are more likely to be poor. Because FRaC does not have the strong self selection property, poor predictors contribute noise to normalized surprisal scores (as shown in Section 1.3.4), so removing high entropy features may improve classification. Negative entropy may thus be a useful filter criterion (as this removes high entropy feature first), though this technique and ordinary low entropy filtering cancel each other out. A function of entropy that filters both high and low entropy features allows them to be used in tandem, such as distance from mean entropy or rank offset from median, would solve this issue.

Technique Combination

Because the various filter criteria outlined above operate on different assumptions, they are quite compatible with ensemblification. For example, as discussed above, KL divergence filtering can identify with certainty excellent candidate features (highly divergent features), but low scores are not particularly meaningful. Conversely, entropy can identify *poor* candidate features (high or low entropy features), but high scores are not particularly meaningful. Ideally, an ensemble using these techniques is capable of identifying both good and poor candidate features, and filtering accordingly.

In order to ensemblify a set of filter criteria, a manner by which to combine their scores is necessary. In the previous example, we have domain knowledge of how the two filter scoring functions operate, and it makes sense to create a function that will assign low scores to features with low KL divergence, low scores to features with low entropy and moderately low KL divergence, and higher scores to other features. However, in the general case a technique for combining arbitrary score vectors is required.

A safe option for criterion combination is to normalize the output vector of each scoring function, and then simply sum them. Vector addition exhibits monoid structure, so this technique is applicable to an arbitrary number of filter criteria, and no one criterion can dominate the scores, as each score vector is normalized. Formally, the technique is as follows:

$$\text{sumscore}(\hat{\mathbf{f}}) = \sum_{\vec{i} \in \text{nonzero}(\hat{\mathbf{f}})} \frac{\vec{i}}{\|\vec{i}\|}$$

where $\hat{\mathbf{f}}$ is a vector of feature score vectors, and nonzero($\hat{\mathbf{f}}$) is the function that takes a list of vectors onto the same list, minus any zero vectors. Alternatively, each vector could be converted to a rank ordering, and the rank scores for each feature for each criterion could be summed.

An alternative to sum combination is *product combination*, where the product of all scores for a particular feature is used as the feature's criterion score. This method is extremely sensitive to low feature scores, as a single low feature score can result in an overall low score. Whether this is desirable depends on the component feature criteria, so this may or may not be a liability. On the other hand, the product technique poorly handles negative feature values, so care needs to be taken when such values are a possibility.

Just as the choice of feature filter criteria depends on the dataset in question, the choice of filter criterion combination technique is also a nontrivial decision. The properties of a dataset, as well as the properties of component criteria, need to be considered when selecting a filter criterion combination technique.

Overview of Filter Techniques

The KL divergence metric assigns features that are known to be good anomaly detection features high values. This contrasts sharply with the entropy metric, which assigns features that are known to be poor anomaly detection features low values. Due to the ideological differences between the two techniques, they should theoretically work quite well in tandem, as features that perform well by both metrics are known to be valuable features, and features that perform poorly by both metrics are known to poor features.

The correlation based metric, although more costly, can detect features that are valuable in the context of other features, which is impossible for the KL divergence and entropy techniques. However, the lack of support for discrete features is a major drawback to this technique.

These simple techniques in tandem should allow for the efficient selection of candidate features, though more complicated techniques may be necessary to discover the subtle relationships between features. Calculating a subset of features on which to run FRaC without losing accuracy is a difficult problem, as the effects of removing any feature are difficult to predict. Solving these problems, finding a balance between time spent selecting features and time saved running FRaC on a reduced feature space, and calculating feature subsets on which FRaC is actually more accurate than over the whole space are the ultimate goals of these feature selection techniques.

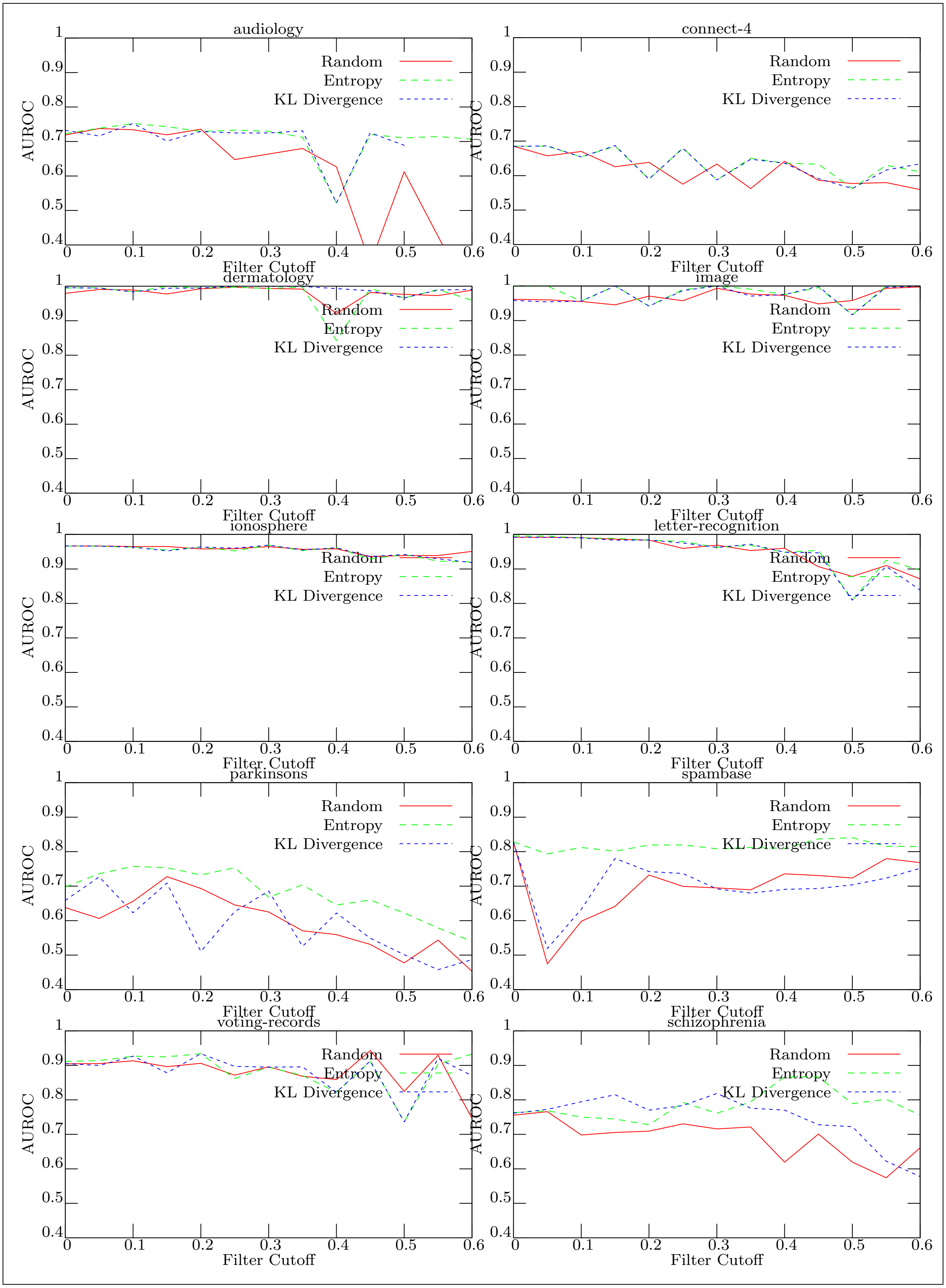


Figure 3.3: Filter Cutoff vs. AUROC Curves, for Various Datasets and Filter Criteria.

3.2.3 Empirical Evaluation

Full Filtering

Presented here are the results of running FRaC with various filter types over several UCI datasets, and the schizophrenia dataset. Plots of the response curves of filter cutoffs against AUROC values for the unsupervised anomaly detection problem are presented in Figure 3.3, and a summary of the data is available in Table 3.1.

The five UCI datasets with the highest feature counts from prior experiments were chosen for this experiment. They were `audiology`, `connect-4`, `dermatology`, `image`, `ionosphere`, `letter-recognition`, `parkinsons`, `spambase`, `voting-records` and `schizophrenia`. The schizophrenia dataset was assembled from genome sequencing data in several studies. Normal instances represent genomic data from healthy subjects, and anomalous instances represent genomic data from schizophrenic individuals. The dataset has 1965 ternary features. More information on this dataset is given in the next section.

Random filtering is also presented, where rather than selecting features to remove by some filter criterion, they are selected randomly. Because the efficacy of randomly selected feature subsets can be highly variant, especially for small subsets, in these experiments the results are averaged over three runs. The random feature selection technique is used to provide a fair baseline against which to compare various filter criteria. A filter criterion is working if it is more accurate than random removal, and a filter criterion is doing something actively wrong if it is less accurate than random removal.

From the summary table, we see that entropy based feature selection is the clear winner, though KL divergence is not far behind, and in some datasets they are quite close. Both techniques certainly outperform random feature selection.

Also interesting is the fickle nature of the response curves in Figure 3.3; one might expect to see accuracy sharply decline as the number of features decreases, but this is often not the case.

Cutoff	Random	Entropy	KL Div
0.05	0.8056	**0.8592**	0.8227
0.1	0.8169	**0.8545**	0.8276
0.15	0.8191	**0.8595**	0.85
0.2	0.832	**0.8416**	0.816
0.25	0.8042	**0.8559**	0.8369
0.3	0.8317	**0.8374**	0.8335
0.35	0.7971	**0.8453**	0.8151
0.4	0.7854	**0.8029**	0.7721
0.45	0.761	**0.85**	0.8068
0.5	0.7586	**0.7895**	0.7549
0.55	0.8024	**0.8282**	0.7958
0.6	0.7234	**0.8136**	0.7853
Overall	0.7948	**0.8365**	0.8097
Mean Rank	2.6923	**1**	2.3077
Maximal Count	0	**13**	0

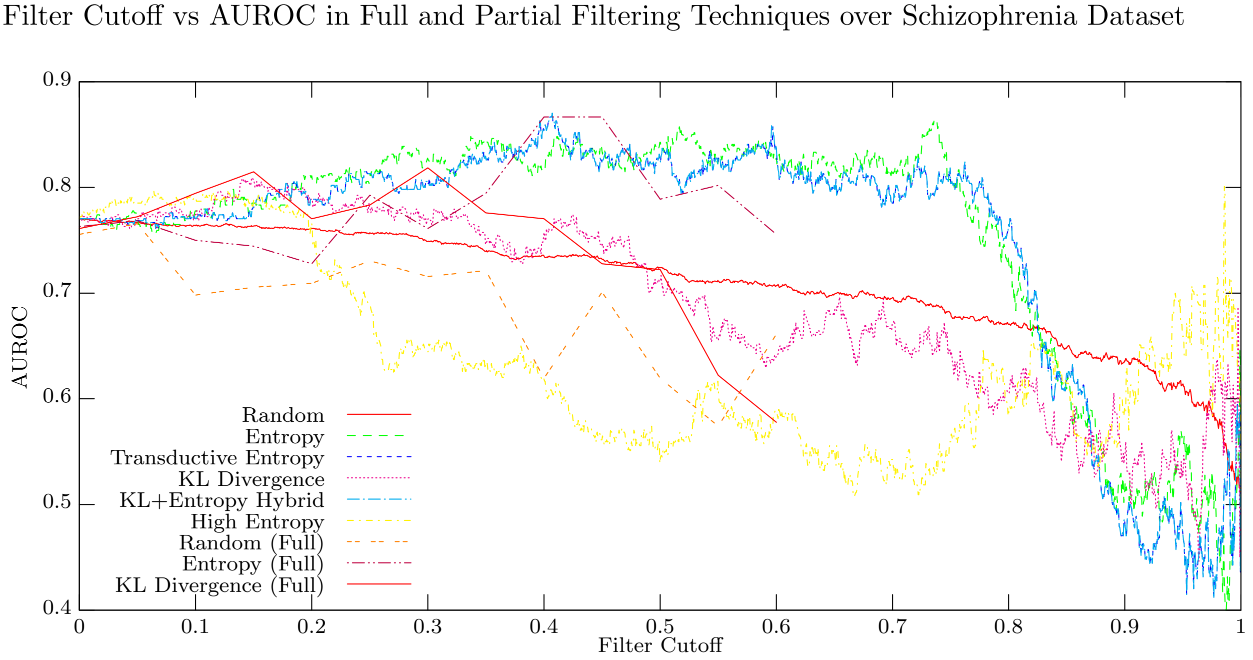
Table 3.1: Comparison of various filter criteria: data summary.
Average performances over the five largest UCI datasets examined in this paper. *Random* refers to random feature selection averaged over 3 runs, *Entropy* refers to non-transductive low entropy filtering, and *KL Div* refers to low KL divergence filtering.

Even with random feature selection, often the performance of FRaC degrades only marginally with 50% feature removal. This is surely a property of the datasets in question, and further research is needed to determine why this happens.

Partial Filtering

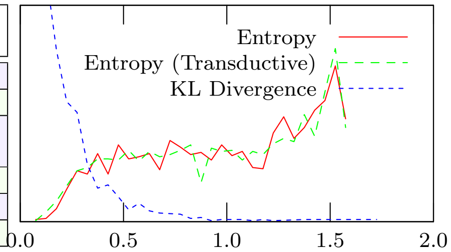
In this section, I present experiments over two datasets to demonstrate the efficacy of partial filtering. Both were picked because of their domain (computational biology) and large feature space (over 1000 features), as filter methods have historically performed well in similar experiments.

Because partial filtering uses the predictors trained on *all* other features, it is possible to use the same predictors (and therefore feature-wise normalized surprisal values) for each fea-



Partial Filtering Criterion	Area Under Performance Curve
Random	0.7127
Entropy	0.7613
Transductive Entropy	0.7529
KL Divergence	0.6940
Hybrid Sum	0.7529
High Entropy	0.6335

Area under AUROC-filter cutoff curves.



Distributions of filter scores.

Here entropy, transductive entropy, KL divergence and hybrid filter criteria are compared to a random feature selection baseline. Full filtering curves are also shown for comparison.

Figure 3.4: Exploration of Partial Filtering over Schizophrenia Dataset

ture of each sample, because the features that predictors are trained on is invariant of the filter scores. By using precalculated normalized surprisal values, it is possible to evaluate FRaC on arbitrary partial filterings for a dataset after only once training models and classifying a test set. This technique was utilized in the following experiments, and because of it, data is available at the finest possible granularity.

KL divergence and entropy based filter criteria, introduced in Section 3.2.2, are investigated. Unless otherwise noted, entropy based filtering refers to removing low entropy features first, though high entropy filtering is explored as well. *Transductive entropy* refers to the technique where the entropy of a feature across the training and query sets is used as a filter criterion, rather than just the training set entropy. Transductive entropy and KL divergence filtering *always* filter low scoring features before high scoring features. Additionally, hybrid techniques between transductive entropy and KL divergence filter scores, including summation and multiplication of these scores, are explored. All filter criteria are compared to a random baseline, established by randomly selecting a subset of features over which to partially filter. To reduce the variance of the curves shown in the diagram, the random baseline curve was averaged over 40 runs.

Schizophrenia The schizophrenia dataset was assembled from multiple studies in the HapMap project (Gibbs et al., 2003). The training set (non-schizophrenic) came from (Redon et al., 2006), GEO accession number **GSE5173**, non-schizophrenic test set instances from the controls in (Scholtysik et al., 2010), GEO accession number **GSE21597**, and the schizophrenic test set instances came from (Vrijenhoek et al., 2008), GEO accession number **GSE12714**. All data was collected using an Affymetrix Mapping 250K Nsp SNP array, which classifies each single nucleotide polymorphism (SNP) as homozygous dominant, homozygous recessive or heterozygous. There is also a small possibility of classification failure, which is interpreted as a missing value when it occurs. The data were then preprocessed to only contain SNPs in protein coding regions, bringing the dataset down to 1965 ternary features. The task was then to identify samples from schizophrenia patients as anomalous. Due to the extreme heterozygosity of schizophrenia, diagnosing it is very difficult even in clinical settings, so overall AUROC values are expected to be low.

In Figure 3.4, the partial filtering technique with various filter criteria are compared. Full filtering results are shown as well, for comparison.

Note that the (low) entropy and transductive entropy filter criteria are overall the highest scoring. Judging by the area under the AUROC vs filter fraction scores, it seems as though the non-transductive variant is superior, however the difference is minor and may be due to chance. Theoretically speaking, the transductive variant has more information, but it may just be that the cases where transductive entropy is useful are not present in this feature set. At a filter cutoff of around 82.5%, techniques drop below the performance of random feature choice, most likely because at this point only high entropy features remain.

It is also worth noting that the high entropy filter does very well early on, beating all other partial filtering criteria until a filter cutoff of about 12%. Presumably, this means that high entropy features are difficult to predict, and thus predictions for them contribute noise. The accuracy of the high entropy filtering technique plateaus around 5%, so presumably the top 5% highest entropy features generally have poor predictors. The scores drop sharply at around 20%, probably signifying that these features are mostly predictable, and soon only low entropy features are left.

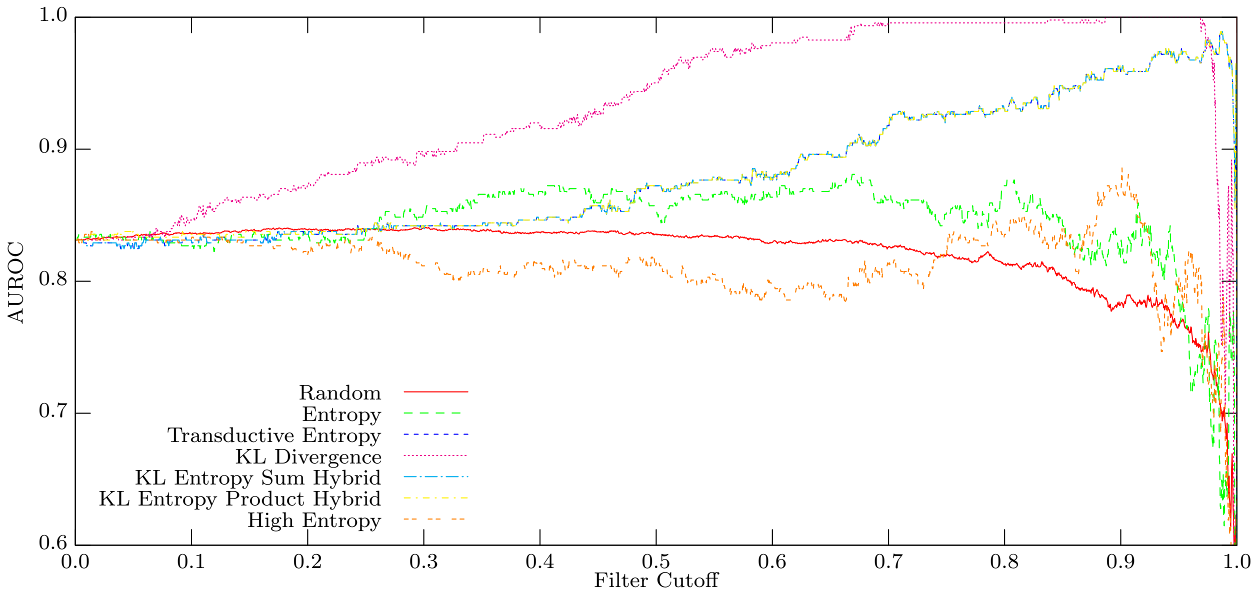
It is interesting to note that both the high and low entropy filtering techniques do well for low filter cutoffs, but eventually drop below the performance of random selection. This supports the hypothesis that both high and low entropy features produce poor predictors, so ideally *both* would be filtered out.

KL divergence scores very poorly overall, though for conservative filter cutoffs, the technique does quite well. This in itself is surprising: since entropy filtering identifies poor features, it should be able to pick them out early on and do well, but instead KL divergence has the lead. Also of note is that at the tail of the graph, KL divergence scores rise sharply. As KL divergence filtering can select very good features (those that are highly divergent), these are the final remaining features, so it makes sense that KL divergence filtering would perform well for high filter cutoff values.

Another notable feature of the response curves are the sharp drops of the low entropy based techniques, beginning with a filter cutoff value of ≈ 0.775 and at ≈ 0.45 in the KL divergence technique. These drop to well below the performance of random feature selection, and for transductive entropy, even drop below $\frac{1}{2}$ over most of the region $[.9, 1]$. These results are difficult to interpret, and it remains to be seen if they hold over other datasets.

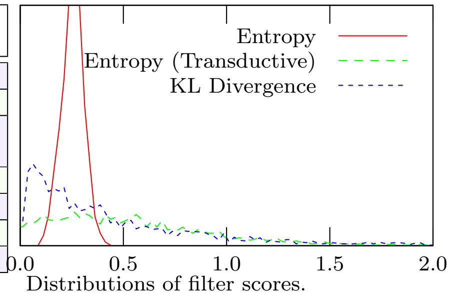
Interestingly, in the KL divergence transductive entropy hybrid technique, performance is nearly identical to that of transductive entropy alone. I hypothesize that this is the case because of the distributions of the two features: nearly every feature has low KL divergence, so these features make the dominant contribution for KL divergence, whereas the entropy of features is more spread out. Ideally, this technique would virtually guarantee that features

Filter Cutoff vs AUROC in Full and Partial Filtering Techniques over Lymphoma Dataset



Partial Filtering Criterion	Area Under Performance Curve
Random	0.8222
Low Entropy	0.8427
Low Transductive Entropy	0.8805
KL Divergence	0.9316
Hybrid Sum	0.8805
Hybrid Prod.	0.8812
High Entropy	0.8128

Area under AUROC-filter cutoff curves.



Distributions of filter scores.

Here entropy, transductive entropy, KL divergence and hybrid filter criteria are compared to a random feature selection baseline.

Figure 3.5: Exploration of Partial Filtering over Lymphoma Dataset

with high KL divergence between training and test set get used, which should improve performance, however the area under the filter cutoff vs AUROC curve shows a (positive) improvement of less than .00005 over that of transductive entropy alone. This is hardly a significant result, and provides evidence against the hypothesis that these two filter techniques operate well in tandem, though this is only one dataset.

A final note is that, as expected, the partial filtering curves for the most part lie above the corresponding full filtering curves. The main exception to this trend is that the full filtering KL divergence curve seems to take on higher values than the partial filtering curve for a brief interval in the beginning. One possible explanation is that some of the features filtered early in the filtering process were so irrelevant that they served only to confuse predictors of other features, so by removing them entirely, better predictors were trained. Due to the low granularity of the full-filtering curve[3], it is difficult to draw a conclusion about a small region.

[3] Running full filtering for all possible filter values would have resulted in over 1000 experiments for each filter type, each taking an average of around 10 hours. However, because the models trained in partial filtering do not depend on whether other features are filtered, the partial filtering experiment only required one run of FRaC where termwise surprisal scores were output. Different combinations of surprisal scores were then added together to generate the curve shown in Figure 3.4.

Lymphoma The *lymphomas* dataset, taken from the CSAX microarray compendium (Noto et al., 2014), consists of 2364 real valued features, each representing the expression level of a particular gene. These values were obtained from a lymphoma specific RNA microarray (Alizadeh et al., 2000). On this dataset, the task is to identify diffuse large B-cell lymphoma given training data consisting of follicular lymphoma and chronic lymphocytic leukemia.

The most striking difference between the lymphoma experiment and the schizophrenia experiment is that on this dataset, KL divergence is by far the best feature criterion. Although this is dataset dependent, a confounding factor is that in this experiment, the test set consisted of 6 normal samples and 77 anomalous, so divergence between the training and test set is very close to divergence between normal and anomalous classes. In the schizophrenia experiment, there are 10 normal samples and 54 anomalous, so the divergence between training and test is less a proxy for that between normal and anomalous. That being said, as the size of the training and query sets increase while the proportion of anomalous samples is held constant, the divergence of a feature between them becomes less subject to sampling bias, so even a tiny fraction of anomalous samples can be detected given sufficient data.

Another significant difference is that the overall AUROC scores for the lymphoma experiment are far higher than those of the schizophrenia experiment. This is an indication that it is a much easier classification problem, or at least one to which FRaC is better suited. It is interesting to note that with random feature selection AUROC values fluctuate on a narrow band occupying around $[0.82, 0.84]$ as filter cutoff scores occupy $[0, 0.70]$. Another way to look at this is that predictors for just 30% of the features are enough to predict whether a sample is anomalous with the same accuracy as the full ensemble. One likely interpretation is that there are some number of highly predictive features, and samples of 30% or more of the features are extremely likely to contain enough of these highly predictive features to enable highly accurate classification. Certainly high performance of small random subsets of features is a dataset dependent property, but it suggests that in many cases, FRaC does far more work than is necessary, and perhaps much of this work has little or no effect on accuracy.

Additionally, the poor performance of high entropy filtering may be an indicator that there are very few difficult to predict high entropy features. This is corroborated by the tight clustering of entropy values around 0.25 bits in the distribution of filter scores. However, it is also important to note that the performance of the high entropy filtering technique overtakes that of random selection on most of the $[0.75, 1]$ interval. This result suggests that on this dataset, some of the low entropy features are actually useful.

In addition to the high accuracy of the random feature subsets, the performance of some of the filter criteria far outstrips that of random feature selection. We see that for filter cutoffs in $[0.89, 0.96]$, the AUROC values of the KL divergence partial filtering curve are nearly perfect. Additionally, the transductive entropy and the hybrid techniques are extremely accurate, even outperforming KL divergence alone for filter cutoffs in excess of 0.975. extremely high cutoff values. All of this suggests that partial filtering is an effective technique, not only for reducing the amount of necessary computation, but also for increasing the accuracy of FRaC.

The performance of the hybrid techniques is a bit underwhelming: for the most part both the hybrid product and hybrid sum techniques are equivalent to the transductive entropy technique, though the hybrid product is slightly more accurate. Ideally hybrid techniques would outperform the component techniques, rather than have intermediate performance. It remains to be seen whether the effect of hybridization between less similar filter criteria will be greater.

Overview Overall, if we assume that these results generalize to other datasets, we can conclude that partial filtering, although computationally more expensive, generally produces more accurate results for the same filter cutoff than does full filtering, though unlike full filtering, partial filtering can not prevent irrelevant features from confusing other models. We also see that at times, depending on choice of filter criterion and filter technique, both partial and full filtering can actually improve accuracy. Full filtering can improve the predictors for unfiltered features (though it can also be detrimental), and partial filtering can improve results by removing surprisal values from poorly scoring features from the equation. Because FRaC does not have the self selection property (see Section 1.3.4), these features contribute noise to normalized surprisal, so removing them leads to greater accuracy.

One question that remains is why the KL divergence technique performs better than random selection for most low filter cutoff values but drops below the performance of random selection for a significant region on the schizophrenia dataset. This effect is not seen the lymphoma experiment, so it may be an artifact of the dataset. Additionally, the fact that high entropy filtering is more accurate than any other criterion for high filter cutoffs in the schizophrenia dataset implies that low entropy features *are* good identifiers of anomalies, which is paradoxical because the efficacy of low entropy filtering implies the opposite. Further research is needed to discover patterns in these effects and elucidate their causes. Furthermore, only two datasets are analyzed here, and it would be dangerous to draw hard conclusions about the general technique from such an analysis.

3.2.4 Filter Method Extensions

Domain Knowledge Integration

Sometimes we have additional knowledge about our features, and the expectation that some will be better predictors of anomality than others. With this additional information, we can perform a modified filter step that combines prior information with an established filter score.

There are many paths one could go down, but for the sake of simplicity, suppose we assign each feature a *prior knowledge* score, where 0 is neutral, high values represent putative valuable features, and lower values represent features that are thought to be of less value to the anomaly detection problem. We then add the prior knowledge score of each feature to its filter score, perhaps first normalizing the filter scores, to obtain a final score for each feature, upon which we then perform the filter step as usual. Using this technique, we can "suggest" to the filter algorithm that some features are probably worth keeping, and others are probably not, but the final decision can be made with the additional information provided by the filter scoring algorithm.

This technique is valuable in several cases. First, some features could be thought to be more relevant to the class than others. For instance, we could try to predict a person's weight by height and gender, because our prior knowledge dictates that these are related, but any link between, say, hair and eye color, could be much more tenuous. This is particularly useful when some features are suspected to be relevant, and some are not, but the truth about each individual feature is not known.

A second use case is when some features come from noisy measurements, in which case we can discount them with the prior knowledge score. This allows us to discriminate against features that are known to be unreliable, except in the case that they should prove themselves to be valuable through the filter score technique.

Domain knowledge is often utilized in dataset creation. In text classification, common words, known as *stop words* (Luhn, 1957), such as "the," "and" and "a" in English, are often ignored or not counted: in essence they have been explicitly filtered based on domain knowledge. Similarly, in personalized medicine, when building models, often a small number of genes thought to be important to the task at hand are used, and the remainder are implicitly filtered. Sophisticated techniques, such as those in (Kim and Bredel, 2013) have been used to select features in these contexts, but such techniques can not be optimal without knowledge of the algorithm that will later operate over the filtered dataset, because two different algorithms may have different optimal filterings *for the same dataset*. For this reason, I advocate for sharing this domain knowledge with the algorithm, rather than using it to filter and then running the algorithm on the filtered dataset[4].

The prefiltering technique has two drawbacks. The first is that domain knowledge can not be combined with other filter criteria, so unexpected relationships can't be identified. For example, in the computational biology space, if a gene is removed by hand *before* filtering, even if it would score highly by some filter criterion, it can never be used in a model. By combining a priori knowledge of a dataset with filter criteria, strong patterns can speak for themselves, while weaker ones can be filtered. The second drawback is that this technique is totally incompatible with partial filtering. By instead making domain knowledge explicit to the feature selection stage, partial filtering and filtering can proceed separately and in an intelligent manner, using filter criterion evidence in place of the decisions made by an operator.

[4] Although I advocate for this technique, I did not in fact use it in the experiments on the schizophrenia dataset, where I removed all noncoding SNPs from the original dataset. This was done for simplicity and tractability, because I was examining only partial filtering.

Chapter 4: Conclusion

4.1 Overview

In this document, I define and evaluate many variants on the FRaC algorithm. Some of them appear to have been more successful than others, though everything I report seems to have a niche of some sort. The beauty of FRaC is that it is a self selecting algorithm capable of introspection, and for this reason hyperparameterization is minimally important. Here I discuss the niche carved out by each of the variants discussed in this paper in a broader context.

Entropy Normalized FRaC eFRaC is essentially a feature weighting algorithm, conceptually related to normalization of distance based algorithms, although not the same thing: as shown in Section 1.3.4, FRaC is normalization insensitive under circumstances in which general algorithms are not. Essentially, in eFRaC the contributions of different features are reweighted with the goal of balancing their contributions. It may be that the features are already implicitly weighted in such a way that reweighting degrades performance, but I provide empirical evidence to show that this is rare in real world datasets (with the dermatology dataset being a notable exception). In most cases, eFRaC improves performance, though the difference is small.

Combinatorial Ensemble FRaC cFRaC is an algorithm developed to generalize the concept of inner and outer ensembles, and combine their output with that of intermediate ensembles. For ensembles of size one, cFRaC is identical to ordinary FRaC, and for ensembles of size two, cFRaC is equivalent to an ensemble of size three, consisting of the two original predictors, and an inner ensemble composed of both of them. For larger ensembles, the results are more sophisticated. cFRaC eliminates the hyperparameterization decision between inner and outer ensembles, is efficient, and empirically seems to outperform both inner and outer ensembles over real world datasets, though the significance of these results is debatable.

Missing Value Corrected FRaC Overall, mFRaC is a very conservative modification, and only applies to datasets with missing values. It appears to be advantageous on some datasets, and not on others. More research is needed to determine under which circumstances the missing value correction of mFRaC increases accuracy.

Filter Methods Filter methods provide a much needed computation time reduction to FRaC, though it can be difficult to predict what effect filtering will have on accuracy. Accuracy often increases as filtering is applied, especially with partial filtering, which is notable because these techniques increase accuracy while reducing runtime.

Analysis from a Feature Modelling Perspective In (Noto et al., 2012), the authors identify three questions which any feature modelling algorithm need address, either explicitly or implicitly. The questions are as follows:

1. For which features do we learn a predictive model, and which other features do we use in each model?

2. Which supervised learning algorithm(s) do we use to train the feature models?

3. How do we combine the set of feature models $\mathcal{C}$ into a single anomaly score for a query instance $\vec{x}_q$? That is, what is the definition of $f(\mathcal{C}, \vec{x}_q) \to \mathbb{R}$?

mFRaC and eFRaC both address question three, providing simple alternatives to the summation in traditional FRaC. cFRaC also addresses question three, albeit in a more sophisticated manner, and to some extent addresses question two, in that the ensembles of cFRaC are themselves learning algorithms.

On the other hand, filtering predominantly addresses question one. In full filtering, we select both features with which to learn models and features to use in these models, and in partial filtering, we only select which features to use in a model.

Experimental Results Many of the results outlined here show that one algorithm seems to outperform another by a small margin. At times, it is difficult to determine whether a result is significant, and running the algorithms over more datasets would make the results more significant. In an attempt to show the significance of these results, when algorithms were compared over a group of datasets, in addition to the mean AUROC, the average rank of each algorithm, as well as the number of times it scored a maximal value, were shown.

4.2 Further Work

I have created several variants of FRaC in this document, and they fit into two broad categories. The first category consists of mathematical modifications to the algorithm. This class essentially represents a new idea of what an anomaly is, and comes with an associated anomaly score formula. The development of normalized surprisal from average probability (used in CFA) is such an example, as are the formulae used in eFRaC and mFRaC.

The second broad category is the group of algorithms that generalize a decision, reducing the number of hyperparameters, allowing FRaC's validation phase to make such decisions. An example of such a technique is cFRaC, which eliminates the choice between inner and outer ensembles, and allows cross validation to adjust the weight of the contributions the inner and outer ensembles (as well as the intermediate ensembles). These techniques are very interesting because they (potentially) allow a broader class of anomalies to be detected while simultaneously reducing the hyperparameterization space.

Most of the following ideas fit into one of these categories.

4.2.1 Alternative Filter Criteria

Although they are discussed briefly in Section 3.2.2, I never actually implemented the correlation based filter or the Jensen-Shannon filter. The correlation based filter has a higher time complexity in terms of the number of features than the others, but it also measures something very different, and is the only filter based technique mentioned in this paper that is capable of taking interfeature relationships into account (due to its $\omega(n)$ complexity). As Jensen-Shannon divergence and KL divergence are similar in many ways, I expect their utility as filter criteria to be similar as well, though considering how effective a filter criterion KL divergence proved to be, one wonders if Jensen-Shannon divergence would be similarly successful.

Also not evaluated were the hybrid filter techniques. The different filters usually measured different heuristic correlates of utility, and were based on diverse assumptions, so it stands to reason that a filter ensemble would yield many of the benefits of ensemblification.

In addition to experiments with these alternative filter criteria, it remains to be seen how much the results I did find generalize. I only ran filtering on a few biological datasets and UCI datasets with 16 or more features. Filter methods traditionally work best in high dimensional spaces, and we may see stronger results with such datasets.

4.2.2 Diverse FRaC

In this work, I presented many minor variations on the basic FRaC theme. Many of them showed minor improvements to the baseline of the original FRaC algorithm, and focused on specific aspects, such as missing value correction, ensemblification, or feature selection. I believe that the next major leap in feature modelling based anomaly detection techniques will need to combine all these advances into something greater than the sum of their parts. I present a candidate here.

In order to take full advantage of the diversity hungry combinatorial FRaC, a large number of

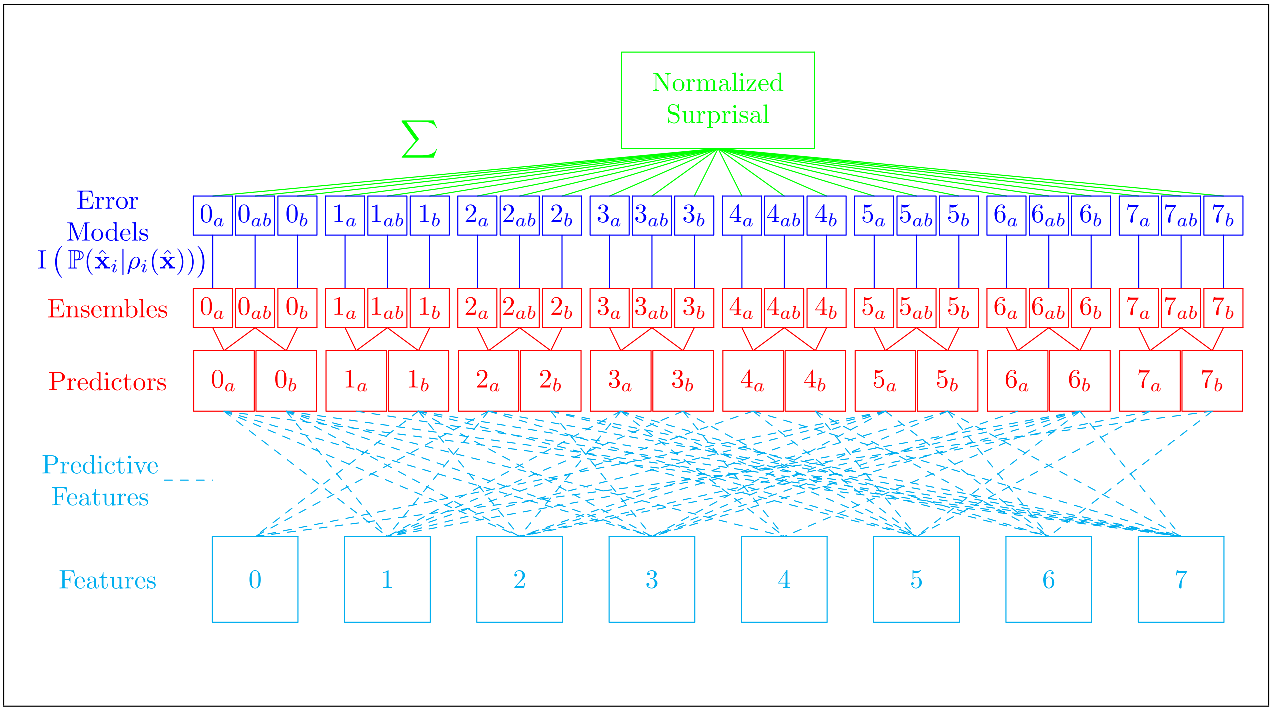

Normalized Surprisal
$\sum$
Error Models
$\mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i|\rho_i(\hat{\mathbf{x}}))\right)$
0_a 0_{ab} 0_b 1_a 1_{ab} 1_b 2_a 2_{ab} 2_b 3_a 3_{ab} 3_b 4_a 4_{ab} 4_b 5_a 5_{ab} 5_b 6_a 6_{ab} 6_b 7_a 7_{ab} 7_b
Ensembles
0_a 0_{ab} 0_b 1_a 1_{ab} 1_b 2_a 2_{ab} 2_b 3_a 3_{ab} 3_b 4_a 4_{ab} 4_b 5_a 5_{ab} 5_b 6_a 6_{ab} 6_b 7_a 7_{ab} 7_b
Predictors
0_a 0_b 1_a 1_b 2_a 2_b 3_a 3_b 4_a 4_b 5_a 5_b 6_a 6_b 7_a 7_b
Predictive Features
Features
0
1
2
3
4
5
6
7

Figure 4.1: Diverse FRaC

classifiers for each feature need be trained. In cFRaC, a group of classifiers is trained for each feature, but each uses all available training data and all features (except the predicted feature). A more sophisticated approach would be to train a number of models, using a random learning algorithm with a random subset of the training data (as in (Breiman, 1996)) and a random subset of the features (as in (Sutton et al., 2005)). Though it is a dataset dependent property, the AUROC response curves in 3.3 suggest that strong predictive models can still be built even with only small subsets of the feature space, and those in 3.4 show that not all features need to be predicted to produce a powerful anomaly detector.

These models would be very diverse, because they would be trained on different data, and therefore base their predictions on different information. Furthermore, any poor predictors trained in this manner would make a minimal contribution to normalized surprisal due to introspection (See the Self Selection Property, in Section 1.3.4). Combinatorial FRaC is capable of using ensembles in a sophisticated manner, and would be able to effectively combine the results of a large number of diverse learners.

Both target and predictor features could be chosen uniformly, though this is a great opportunity to utilize some of the filter criterion research. By running a feature scoring algorithm, a score can be produced for each feature. These scores can be used (transforming such that they are nonnegative if necessary) as weights, and feature selection could be performed by weighted, rather than uniform, sampling. This favors certain features that are thought to be better anomaly identifiers over others, but is more conservative than filtering or partial filtering.

Although elegant and simple, this technique is not without its problems. Cross validation in FRaC is already somewhat suspect, as the error model was trained across a collection of smaller models based on subsets of the training data rather than the final model. With this technique, where model selection is random and on random subsets of the data, cross validation is even more questionable. However, because a random subset of samples is used for model construction, the remainder could be used for error model collection, possibly along with a smaller validation set, so cross validation may not be necessary.

A second issue is that as the number of models for a feature increases, the total number of summed values increases *at an exponential rate.* Because different features have different numbers of predictors, one feature could be represented significantly more in the final outcome. Such a property violates the empirically supported conjecture of eFRaC: different features should make similar overall surprisal contributions. This can be accounted for by multiplying each feature term by $\frac{1}{2^{\hat{\mathbf{M}}_i - 1}}$, for $\hat{\mathbf{M}}_i$ the number of models trained for feature i (and treating possible $\frac{0}{0}$ values as 0).

This technique very much fits into the category of hyperparameter-reducing algorithms, as it folds the benefits of partial *and* complete filtering into the benefits of cFRaC, *without* requiring the setting of a specific filter cutoff. Diverse FRaC is interesting because it provides sophisticated answers to feature modelling questions one and three of (Noto et al., 2012), establishing a sophisticated manner of selecting features for which to train models and features to use in said models as well as combining feature scores using the ensemblification technique of cFRaC. For a visual diagram of Diverse FRaC, see 4.1.

4.2.3 FRaC and Conditional Probabilities

Though some interesting derivations were made for cFRaC, it is not clear that the assumptions made were significantly more valid than those in the original FRaC definition. It seems as though more complicated anomaly scores could be developed using conditional probabilities of the output of various classifiers for various features. Such techniques would provide a significantly more sophisticated answer to the third question of feature modelling techniques as proposed by Noto et al. in (Noto et al., 2012) . Any sort of refinement of the evidence space could yield more informative events and potentially more information-dense surprisal values for situations that current iterations of FRaC can not distinguish. Some of the logic in (Read et al., 2011), based on CFA, may be reframable in terms of FRaC, and may yield to smarter conditional probability calculations.

4.2.4 FRaC and Probabilistic Predictors

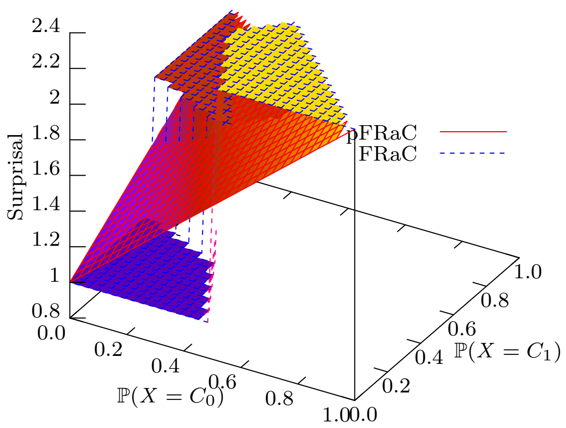
Many classifiers are capable of outputting a probability distribution over possibilities, rather than a single classification (indeed, many work this way internally, and output the maximum-likelihood value). We can form a reduction hierarchy of classifiers, where ordinary classifiers $\leq_m$ classifiers with confidence $\leq_m$ probabilistic classifiers. Similarly, some regressors are capable of outputting a distribution, though the concept of a regressors with confidence values is more complicated, due to the nature of continuous probability spaces.

Both classifiers with confidence values and probabilistic classifiers give more information than an ordinary classifier, but even when this information is available, it is ignored by FRaC. This means that FRaC treats a very certain classification, and one that is, say, 49% to 51%, identically. This results in instability around these uncertain classifications, and furthermore the effects of very certain classifications are diluted by the effects of uncertain classifications.

To remedy this issue, I propose the pFRaC algorithm, which calculates the *expected normalized surprisal* in place of normalized surprisal. The definition is as one would expect:

$$
\begin{aligned}
&\mathrm{E}[\mathrm{NS}(\hat{\mathbf{x}})] = \\
&\mathrm{E}\left[\sum_{i=1}^{F}\sum_{m=1}^{M}\begin{cases} 0 \text{ if } \hat{\mathbf{x}}_i \text{ is missing, else:} \\ \mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i \mid C_{i,m}(\rho_i(\hat{\mathbf{x}})))\right) - \mathrm{H}(\hat{\mathbf{v}}_i^T) \end{cases}\right] \\
&= \sum_{i=1}^{F}\sum_{m=1}^{M}\begin{cases} 0 \text{ if } \hat{\mathbf{x}}_i \text{ is missing, else:} \\ \mathrm{E}\left[\mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i \mid C_{i,m}(\rho_i(\hat{\mathbf{x}})))\right)\right] - \mathrm{H}(\hat{\mathbf{v}}_i^T) \end{cases}
\end{aligned}
\tag{4.1}
$$

This inner expectation is a bit unusual in that there is a random variable (the classifier prediction) in the evidence space of a probability judgment, but other than that it is a completely normal expectation calculation.



FRaC Surprisal and pFRaC Expected Surprisal by Classifier Probability

$$\begin{array}{cc} & \text{Confusion Matrix} \\ & \text{Predicted Class} \\ \begin{array}{c}\text{True}\\ \text{Class}\end{array} & \begin{bmatrix} .7 & .1 & .2 \\ .1 & .6 & .3 \\ .2 & .3 & .5 \end{bmatrix} \end{array} \quad \begin{array}{cc} & \text{Relevant Column} \\ & \text{Predicted Class} \\ \begin{array}{c}\text{True}\\ \text{Class}\end{array} & \begin{bmatrix} - & - & .2 \\ - & - & .3 \\ - & - & .5 \end{bmatrix} \end{array}$$

X and Y axes represent the probability that the true class, X, is class 0 or 1, respectively, given the evidence. The remainder of the probability mass (1−X−Y) represents the probability that the true class is class 2.

True class is class 3.

Here the outlier scores for a 3 class categorical feature are shown, alongside the confusion matrix for the feature.
Note the planarity of the pFRaC surface contrasted with the discontinuous FRaC surface. Also note the equality between the scores produced by both algorithms on points of certainty.

Figure 4.2: pFRaC vs FRaC Featurewise Outlier (Surprisal) Scores.

Expected normalized surprisal changes smoothly with classifier probabilities, in stark contrast to the discontinuities that result when a nonprobabilistic classifier tips the balance to a new class. This phenomenon is illustrated in Figure 4.2, where we see that, for an individual feature, surprisal changes abruptly as classification probabilities change, whereas expected surprisal is an affine function of the probability of each class.

The use of pFRaC promises to be more robust to the effects of unstable classifiers, which is important in and of itself, but it also has important implications for use with cFRaC. In cFRaC, classifier ties are a problem in that ties lead to instability, as they are randomly broken. With probabilistic classifiers, ensemblification can be done by sampling uniformly from each constituent classifier, and furthermore, with expected surprisal calculations, ties do not need to be broken. The probabilistic version of cFRaC uses the following ensemble score:

$$\mathrm{ECNS}(\hat{\mathbf{x}}) = \sum_{i=1}^{F} \sum_{e=1}^{2^M} \begin{cases} 0 & \text{if } \hat{\mathbf{x}}_i \text{ is missing, else:} \\ \mathrm{E}\left[\mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i \mid E_{i,e}(\rho_i(\hat{\mathbf{x}}))) \right) \right] - \mathrm{H}(\hat{\mathbf{v}}_i^T) \end{cases} \tag{4.2}$$

where here $E_{i,e}$ is as before, except it represents the probabilistic ensemblification of a group of predictors, and thus outputs a distribution (and does not need random tie breaking in the case of discrete variables).

4.2.5 Strong Self Selection

In 1.3.4, the self selection property and the weak self selection property are defined. It is also shown that in the general case, FRaC has neither.

There are obvious solutions to this conundrum, such as using an error model to determine if a predictor is a poor predictor and discarding the predictor if so, which may represent a marginally improved new algorithm, however because the definition of a poor predictor is so specific, it is unlikely that they often occur in practice. A generalization of poor predictors to a continuous metric representing *poorness*, paired with an outlier score that weights each predictor by its poorness would have the strong self selection property, though doing so in a way that is of practical use remains an open problem. Additionally, these concepts have primarily been discussed in the discrete case, and it remains to be seen how well they translate to the continuous. I present a candidate algorithm for the discrete case below.

The following *conditional divergence corrective factor* is to be applied to each predictor's normalized surprisal contribution:

$$\mathrm{Div}_{\mathrm{Cond}}(\hat{\mathbf{f}}, \mathrm{T}, \mathrm{P}) = \sum_{i=1}^{\dim \hat{\mathbf{f}}} \mathrm{D}_{\mathrm{KL}}(\mathrm{T}_{\mathrm{P}=i}(\hat{\mathbf{f}}) \,\big\|\, \mathrm{T}(\hat{\mathbf{f}})) \tag{4.3}$$

where $\hat{\mathbf{f}}$ is the vector of possible values in a categorical feature, $\mathrm{T}(\hat{\mathbf{f}})$ is the true distribution of the feature, and $T_{\mathrm{P}=i}(\hat{\mathbf{f}})$ is the true distribution of the feature given prediction i. I now show a property of the conditional divergence corrective factor.

Proposition 3. *The conditional divergence corrective factor, when applied to the normalized surprisal formula, results in an outlier score with the strong self selection property.*

Proof. To show the strong self selection property, it suffices to show that all poor predictors shall make no contribution to the outlier score, which in FRaC entails a contribution of 0 to the overall sum.

Suppose P is a poor predictor. Then, from equations 1.2 and 4.3, the contribution to normalized surprisal of P is equal to:

$$\begin{cases} 0 \ \text{ if } \hat{\mathbf{x}}_i \text{ is missing, else:} \\ \mathrm{I}\left(\mathbb{P}(\hat{\mathbf{x}}_i \mid C_{i,m}(\rho_i(\hat{\mathbf{x}})))\right) - \mathrm{H}(\hat{\mathbf{v}}_i^T) \end{cases} \cdot \sum_{i=1}^{\dim \hat{\mathbf{f}}} \mathrm{D}_{\mathrm{KL}}(\mathrm{T}_{\mathrm{P}=i}(\hat{\mathbf{f}}) \,\big\|\, \mathrm{T}(\hat{\mathbf{f}})) \tag{4.4}$$

I now denote the left factor α, and note that it is always defined. The right factor is simply the conditional divergence corrective factor. Because P is a poor predictor, it follows that $\forall \in \hat{\mathbf{f}}, \mathrm{T}(\hat{\mathbf{f}}) = \mathrm{T}_{\mathrm{P}=i}(\hat{\mathbf{f}})$. Thus each $\mathrm{D}_{\mathrm{KL}}(\mathrm{T}_{\mathrm{P}=i}(\hat{\mathbf{f}}) \,\|\, \mathrm{T}(\hat{\mathbf{f}}))$ is equal to $\mathrm{D}_{\mathrm{KL}}(\mathrm{T}_(\hat{\mathbf{f}}) \,\|\, \mathrm{T}(\hat{\mathbf{f}}))$, and by the properties of divergences, this is equal to 0. This leaves a final score of $\alpha \cdot \sum_{i=1}^{\dim \hat{\mathbf{f}}} 0 = 0$. Because the corrective factor is identically 0, the product is constant, so the surprisal of any prediction is completely ignored.

Thus it has been shown that the conditional divergence corrected normalized surprisal contribution of a poor predictor is 0, and it follows that this technique has the strong self selection property.

□

This factor is conceptually not unlike the Jenson-Shannon divergence, and is very much like a multary λ divergence, where the λ parameter for each conditional true distribution is controlled by the predicted distribution.

A bit of analysis leads to the result that the expected value on a *perfect predictor* (one such that the normalized confusion matrix is the identity matrix) is as follows:

$$\begin{aligned} &\sum_{i=1}^{\dim \hat{\mathbf{f}}} \mathrm{D}_{\mathrm{KL}}(\mathrm{T}_{\mathrm{P}=i}(\hat{\mathbf{f}}) \,\big\|\, \mathrm{T}(\hat{\mathbf{f}})) \\ &\quad = \sum_{i=1}^{\dim \hat{\mathbf{f}}} \sum_{j=1}^{\dim \hat{\mathbf{f}}} \mathrm{T}_{\mathrm{P}=i}(\hat{\mathbf{f}})(j) \cdot \log \frac{\mathrm{T}_{\mathrm{P}=i}(\hat{\mathbf{f}})(j)}{\mathrm{T}_T(\hat{\mathbf{f}})(j)} \\ &\quad = \sum_{i=1}^{\dim \hat{\mathbf{f}}} 0 + \ldots + 1 \cdot \log \frac{1}{\mathrm{T}_T(\hat{\mathbf{f}})(i)} + \ldots + 0 \\ &\quad = \sum_{i=1}^{\dim \hat{\mathbf{f}}} -\log \mathrm{T}_T(\hat{\mathbf{f}})(i) = \sum_{i=1}^{\dim \hat{\mathbf{f}}} \mathrm{I}\left(\mathrm{T}_T(\hat{\mathbf{f}})(i)\right) \end{aligned}$$

where $\mathrm{T}_x(\hat{\mathbf{f}})(j)$ is the probability of the j^{th} value of the conditional distribution T_T of the feature.

Armed with this equation, we could apply an additional multiplicative factor to the corrective factor such that their product is one for perfect predictors, in which case we have an algorithm reminiscent of traditional FRaC. Alternatively, we could modify the corrective factor to attain the value of the inverse of the feature entropy for perfect predictors, in which case the algorithm would resemble eFRaC.

I end this section with two conjectures about the conditional divergence corrective factor. The first states that the corrective factor is maximal on a perfect predictor, and the stronger second conjecture states that this corrective factor is (strictly) greater on more accurate predictors. I formalize these below:

Conjecture 1. *P a perfect predictor* $\Rightarrow \forall P' \, \mathrm{Div}_{\mathrm{Cond}}(\hat{\mathbf{f}}, T, P) \geq \mathrm{Div}_{\mathrm{Cond}}(\hat{\mathbf{f}}, T, P')$.

Conjecture 2. $\forall T, P, P'$ *defined on* $f, \mathrm{Acc}(P) > \mathrm{Acc}(P)' \Rightarrow \mathrm{Div}_{\mathrm{Cond}}(\hat{\mathbf{f}}, T, P) > \mathrm{Div}_{\mathrm{Cond}}(\hat{\mathbf{f}}, T, P)$, *where* $\mathrm{Acc}(x)$ *represents the overall accuracy of an error model* x.

Both of these conjectures represent desirable traits for a corrective factor of this nature. Like the strong self selection property, these traits represent properties that an introspective algorithm would ideally possess.

4.3 Closing Remarks

Some of the techniques in this document seem promising, though further experimentation is needed to confirm their applicability to real world problems. The techniques proposed in the previous section suggest further avenues for research that build upon the work established here. Exploring alternative FRaC variants in terms of the three questions proposed in (Noto et al., 2012) seems to be an interesting direction to take feature modelling research, as does the pursuit of desirable theoretical properties, such as the strong self selection property.

In the techniques examined here and in the future work section, there is a general trend toward using information more intelligently, rather than obtaining more information. FRaC in its original form can scale up to using an arbitrary number of predictors, and the experiments in (Noto et al., 2012) suggest that more predictors make a better anomaly detector, however the lack of the strong self selection property means that this improvement is limited: regardless of how many predictors are used, each contributes noise on features that it predicts poorly. Techniques like cFRaC use the information produced by predictors more intelligently, rather than simply using more predictors, and techniques like the conditional divergence based corrective factor address the fundamental limitations of FRaC, rather than try to overcome them with more information.

In the future, ideally algorithms will be able to spend less time to make more accurate predictions. Additionally, big data keeps getting bigger, so techniques need to be able to scale up efficiently, both in terms of the size of datasets and the number of features. For these reasons, smarter techniques are an important research direction, as they allow more accurate anomaly detection in less time.

Acknowledgements

First of all, I would like thank my advisor, Professor Donna K. Slonim, for guiding me through the research process and helping me to write my thesis. I also want to thank her for putting so much time and thought into discussing the ideas therein: many of them would not have come to fruition without her input.

I would also like to thank my thesis committee, Professors Anselm Blumer and Benjamin J. Hescott, for their interest in my work, for serving on my committee, for their stimulating discussion on a plethora of related topics, for helping me to write my thesis, and for supporting me throughout the process. Professor Blumer was invaluable in clarifying many machine learning points, and also provided feedback on many mathematical statements that enabled me to make my writing much more precise. Professor Hescott was extremely helpful with matters of protocol, and was instrumental in keeping me on track throughout the process.

On the technical side, I would like to thank the authors of the original FRaC papers, Keith Noto, Carla Brodley, and Donna Slonim, for providing their source code. Their implementation was an invaluable launch point for my own experiments. Additionally, Keith Noto gave me access to the datasets used to test FRaC in (Noto et al., 2012), for which I am quite grateful. I would also like to thank Christopher Michael Pietras for preparing the schizophrenia dataset, which was used in evaluating pFRaC.

I would like to thank Joshua Liebow-Feeser, both for his interest in my work, and our late night discussions on information theory and an eclectic palette of topics. Discussions with Joshua in part led to a resolution of the problems with naïve eFRaC.

Finally, I would like to thank the system administrators at Tufts University for not killing my processes despite their inordinate resource utilization.

References

Ash A Alizadeh, Michael B Eisen, R Eric Davis, Chi Ma, Izidore S Lossos, Andreas Rosenwald, Jennifer C Boldrick, Hajeer Sabet, Truc Tran, Xin Yu, et al. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.

L Douglas Baker and Andrew Kachites McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103. ACM, 1998.

Richard Bellman. Dynamic programming, 1957.

Leo Breiman. Bagging predictors. *Machine learning*, 24 (2):123–140, 1996.

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000. ISSN 0163-5808. doi: 10.1145/335191.335388. URL `http://doi.acm.org/10.1145/335191.335388`.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

William Chauvenet. A manual of theoretical and practical astronomy: Embracing the general problems of spherical astronomy, the special applications to nautical astronomy, and the theory and use of fixed and portable astronomical instruments, with an appendix on the method of least squares, 1863.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16(1):321–357, 2002.

Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119. Springer, 2003.

Frans M Coetzee. Correcting the kullback–leibler distance for feature selection. *Pattern recognition letters*, 26(11):1675–1683, 2005.

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55 (1):119–139, 1997.

Carl Friedrich Gauß. *Theoria motvs corporvm coelestivm in sectionibvs conicis solem ambientivm (Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections)*. Frid. Perthes et JH Besser, 1809.

Carl Friedrich Gauß. *Theoria combinationis observationum erroribus minimis obnoxiae (Theory of the Combination of Observations Least Subject to Error)*. H. Dieterich, 1823.

Richard A Gibbs, John W Belmont, Paul Hardenbol, Thomas D Willis, Fuli Yu, Huanming Yang, Lan-Yang Ch'ang, Wei Huang, Bin Liu, Yan Shen, et al. The international hapmap project. *Nature*, 426(6968):789–796, 2003.

Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

Mark A Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

YA Huang, W Fan, W Lee, and PS Yu. Cross-feature analysis for detecting ad-hoc routing anomalies. *ICDCS '03 proceedings of the 23rd international conference on distributed computing systems*, 2003.

George H John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.

Hyunsoo Kim and Markus Bredel. Feature selection and survival modeling in the cancer genome atlas. *International journal of nanomedicine*, 8(Suppl 1):57, 2013.

Edwin M Knorr and Raymond T Ng. Finding intensional knowledge of distance-based outliers. In *VLDB*, volume 99, pages 211–222, 1999.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, pages 79–86, 1951.

Pierre-Simon Laplace. Essai philosophique sur les probabilits (philosophical essay on probabilities). 1814.

Moshe Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Jianhua Lin. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151, 1991.

Hans Peter Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.

Richard Maclin and David Opitz. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 1999.

Keith Noto, Carla Brodley, and Donna Slonim. Anomaly detection using an ensemble of feature models. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 953–958. IEEE, 2010.

Keith Noto, Carla Brodley, and Donna Slonim. Frac: a feature-modeling approach for semi-supervised and unsupervised anomaly detection. *Data mining and knowledge discovery*, 25(1):109–133, 2012.

Keith Noto, Carla Brodley, Saeed Majidi, Diana W Bianchi, and Donna K Slonim. Csax: Characterizing systematic anomalies in expression data. In *Research in Computational Molecular Biology*, pages 222–236. Springer, 2014.

Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11): 559–572, 1901.

Benjamin Peirce. Criterion for the rejection of doubtful observations. *The Astronomical Journal*, 2:161–163, 1852.

Tomáš Pevnỳ. Anomaly detection by bagging. In *Solving Complex Machine Learning Problems with Ensemble Methods*, pages 25–40. COPEM, 2013.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

Richard Redon, Shumpei Ishikawa, Karen R Fitch, Lars Feuk, George H Perry, T Daniel Andrews, Heike Fiegler, Michael H Shapero, Andrew R Carson, Wenwei Chen, et al. Global variation in copy number in the human genome. *nature*, 444(7118):444–454, 2006.

Karl-Michael Schneider. A new feature selection score for multinomial naive bayes text classification based on kl-divergence. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 24. Association for Computational Linguistics, 2004.

Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

René Scholtysik, Markus Kreuz, Wolfram Klapper, Birgit Burkhardt, Alfred C Feller, Michael Hummel, Markus Loeffler, Maciej Rosolowski, Carsten Schwaenen, Rainer Spang, et al. Detection of genomic aberrations in molecularly defined burkitts lymphoma by array-based, high resolution, single nucleotide polymorphism analysis. *Haematologica*, 95(12):2047–2055, 2010.

Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 40(1):185–197, 2010.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(1): 379–423,623–656, 1948.

Jun Shao. Linear model selection by cross-validation. *Journal of the American statistical Association*, 88 (422):486–494, 1993.

Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. Technical report, DTIC Document, 2003.

Kent A Spackman. Signal detection theory: Valuable tools for evaluating inductive learning. In *Proceedings of the sixth international workshop on Machine learning*, pages 160–163. Morgan Kaufmann Publishers Inc., 1989.

Stephen V Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89, 1997.

Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *J. Royal Stat. Soc.*, 36(2): 111147, 1974.

Aravind Subramanian, Pablo Tamayo, Vamsi K Mootha, Sayan Mukherjee, Benjamin L Ebert, Michael A Gillette, Amanda Paulovich, Scott L Pomeroy, Todd R Golub, Eric S Lander, et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 102(43):15545–15550, 2005.

Charles Sutton, Michael Sindelar, and Andrew McCallum. Feature bagging: Preventing weight undertraining in structured discriminative learning. 2005.

Lena Tenenboim-Chekina, Lior Rokach, and Bracha Shapira. Ensemble of feature chains for anomaly detection. In *Multiple Classifier Systems*, pages 295–306. Springer, 2013.

Terry Vrijenhoek, Jacobine E Buizer-Voskamp, Inge van der Stelt, Eric Strengman, Genetic Risk, Chiara Sabatti, Ad Geurts van Kessel, Han G Brunner, Roel A Ophoff, Joris A Veltman, et al. Recurrent cnvs disrupt three candidate genes in schizophrenia patients. *The American Journal of Human Genetics*, 83(4):504–510, 2008.

David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

Eric P Xing, Michael I Jordan, Richard M Karp, et al. Feature selection for high-dimensional genomic microarray data. In *ICML*, volume 1, pages 601–608. Citeseer, 2001.

Ping Zhang. Model selection via multifold cross validation. *The Annals of Statistics*, pages 299–313, 1993.

Arthur Zimek, Ricardo JGB Campello, and Jörg Sander. Data perturbation for outlier detection ensembles. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*, page 13. ACM, 2014a.

Arthur Zimek, Ricardo JGB Campello, and Jörg Sander. Ensembles for unsupervised outlier detection: challenges and research questions a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):11–22, 2014b.

Index