Substantial Improvements in the Set-Covering Projection Classifier CHIRP (Composite Hypercubes on Iterated Random Projections)

LELAND WILKINSON, Department of Computer Science, University of Illinois at Chicago
ANUSHKA ANAND, Department of Computer Science, University of Illinois at Chicago
TUAN NHON DANG, Department of Computer Science, University of Illinois at Chicago

In [Wilkinson et al. 2011] we introduced a new set-covering random-projection classifier that achieved average error lower than that of other classifiers in the Weka platform. This classifier was based on an L^∞ norm distance function and exploited an iterative sequence of three stages (projecting, binning, and covering) to deal with the curse of dimensionality, computational complexity, and nonlinear separability. We now present substantial changes that improve robustness and reduce training and testing time by almost an order of magnitude without jeopardizing CHIRP's outstanding error performance.

Categories and Subject Descriptors: H.2.8 [**Database Applications**]: Data mining

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Supervised Classification, Random Projections

ACM Reference Format:
Wilkinson, L., Anand, A., Dang, T. N. 2011. Substantial Improvements in the Set-Covering Projection Classifier CHIRP (Composite Hypercubes on Iterated Random Projections. ACM Trans. Knowl. Discov. Data. 0, 0, Article 0 (2012), 18 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

This paper reviews the design of the CHIRP classifier introduced in [Wilkinson et al. 2011] and presents important revisions and extensions of the CHIRP algorithm that substantially improve its performance. CHIRP was designed to address the curse of dimensionality and exponential complexity by using projection, binning, and covering in a sequential framework. For class-labeled points in high-dimensional space, CHIRP employs computationally-efficient methods to construct 2D projections and sets of rectangular regions on those projections that contain points from only one class. CHIRP organizes these collections of projections and regions into a decision list for scoring new data points. The scoring model is based on sets of rectangles, called Composite Hypercube Description Regions.

1.1. Composite Hypercube Description Regions (CHDRs)

While the union of open spherical balls is used to define a basis for the L^2 Euclidean metric topology, we can alternatively use balls based on other L^p metrics. For CHIRP,

This work is supported by NSF/DHS grant DMS-FODAVA-0808860.
Authors' email-addresses: Leland Wilkinson,leland.wilkinson@systat.com; Anushka Anand, aanand2@uic.edu; Dang Nhon Tuan, tdang@cs.uic.edu.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

we employ the L^∞ or sup metric:

$$||x||_\infty = sup(|x_1|, |x_2|, \ldots |x_n|)$$

when we search for neighbors. In this search, we are looking for all neighbors of a point at the center of a hypercube of fixed size in a vector space. Because we are concerned with finite-dimensional vector spaces in practice, we will use $max()$ instead of $sup()$ from now on.

Definition 1.1. A *hypercube description region* (HDR) is the set of points less than a fixed distance from a single point (called the *center*) using the L^∞ norm. A weighted hypercube description region is an HDR that uses the positively weighted L^∞ norm:

$$||x||_\infty = max(w_1|x_1|, w_2|x_2|, \ldots w_n|x_n|).$$

We will assume the term HDR refers to this more general case. Our use of weights implies that different points in a high-dimensional space can have different weights defining their hypercubes.

Definition 1.2. A *composite hypercube description region* (CHDR) is the set of points inside the union of zero or more hypercube description regions.

The CHDR is the structure we use to define a region containing points belonging to a single class or to no class. CHDRs are defined for any number of dimensions in a finite-dimensional vector space. For scalability, we have limited them to two dimensions. The original motivation for working with CHDRs was a visual classifier [Anand et al. 2009] that generated a number of 2D projections and presented them sequentially to users in displays like Figure 1. The subjects constructed CHDRs by drawing rectangular regions around yellow dots on the display. This crude classifier worked surprisingly well. Following those promising results, we subsequently removed humans from the loop and we now grow CHDRs using an iterated covering algorithm. Figure 1 shows an example of a CHDR covering a 2D projection of the Orange10 dataset used in our tests later in this article. The CHDR represented by the blue region of the figure is a composition of rectangles extending out to the edges of the frame.

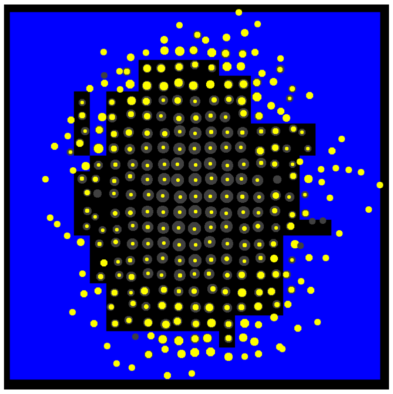

Fig. 1. A CHDR covering class instances at the periphery of a projected random spherical distribution. Data are the Orange10 dataset from [Hastie et al. 2001]. The data have been binned into a 24×24 grid with the size of each dot proportional to the count of instances in each bin. Yellow is used to represent the currently selected class; gray represents all other classes. All-yellow dots represent bins containing only current class instances. All-gray dots represent bins containing only other-class instances. Gray dots with yellow centers and yellow dots with gray centers represent mixed-class bins. The CHDR (a union of rectangles) is colored blue; it covers bins such that the odds of covering current class instances vs. all other instances are maximized.

1.2. CHIRP Constructs a List of CHDRs

Each CHDR constructed by CHIRP is based on a different random 2D projection. The algorithm is a one-against-all classifier [Dietterich and Bakiri 1995]. For each class C_k in a training set, we (a) compute a 2D projection, (b) bin the projected data values in a 2D rectangular segmentation, and (c) cover bins containing mostly instances of C_k with a CHDR. We iterate over classes until we are unable to find bins pure enough to classify remaining instances in the training set.

The result of this process is a list of CHDRs that can be used to score new data points. A point is assigned to the first CHDR in the list that contains it. If no CHDR contains the point, it is assigned to the closest CHDR in the list using smallest point-to-rectangle L^∞ distance.

CHIRP is an ensemble classifier. We run it m times and score a testing instance based on simple-majority, equally-weighted vote. In this paper, we use $m = 7$. Increasing m improves accuracy, but with diminishing gains.

Although CHIRP employs some well-known ideas, the combination of them described in this paper results in a classifier that is novel and coherent. We will first discuss related work, then present the algorithm, and finally present performance comparisons between CHIRP and competitors. We will argue, in conclusion, that the success of CHIRP is due to the statistical properties of its components and the way they are combined.

2. RELATED WORK

Perhaps the most widespread use of rectangular description regions is in recursive partitioning trees [Breiman et al. 1984; Quinlan 1993]. These methods partition a space into nested rectangular regions that are relatively homogeneous over the values of a predicted variable. Our approach differs from these models, however, because it is not restricted to a partitioning. Our description regions need not be disjoint or exhaustive.

Several teams have developed projection-pursuit classifiers [Lee et al. 2005; Flick et al. 1990; Jimenez and Landgrebe 1995]. These efforts exploit the flexibility of affine projections but have failed to ameliorate the computational complexity of projection pursuit.

Researchers have used hyperboxes for classification through neural networks [Simpson 1992], mixed integer programming [Üney and Türkay 2006], set covering [Marchand and Shawe-Taylor 2002], and decision lists [Sokolova et al. 2003; Aguilar et al. 1998]. These approaches can be slow to converge on larger datasets. Most importantly, the hyperbox researchers restrict their method to 2D axis-parallel (pairs of features) projections, so their utility is limited.

Finally, various researchers have used compositions of rectangles (unions and products) to characterize the results of unsupervised classification [Alpern and Carter 1991; Agrawal et al. 1998; Bu et al. 2005; Gao 2002; Gao and Ester 2006; Pu and Mendelzon 2005]. The primary focus of these researchers has been to develop rapid scoring methods that can be implemented inside a database through the use of rectangles. We will discuss some of this work in more detail as we describe the CHIRP algorithm in the next sections.

3. CHIRP TRAINING

The CHIRP training algorithm consists of three stages – *projecting*, *binning*, and *covering*. We will describe these stages in detail in this section. First, however, we will summarize preliminary data processing steps similar to those employed in other classifiers.

3.1. Preliminary Normalizing and Transforming

We begin by reading n rows and p columns of a training dataset X. We code numerical values as double precision numbers and string values as integers. We assume numerical values are derived from *continuous variables* and string values from *categorical variables*, although numerals can be treated as strings if so designated. We use the terms *feature* and *variable* interchangeably to mean a mapping of a set of objects to a set of values.

We normalize the data by rescaling each variable (feature) to the open unit interval. Then we recode extremely skewed variables with a nonlinear transformation. In the KDD paper, we used a folded square root transformation. Our decision on whether to apply this transformation to a given variable was based on computing conventional standardized skewness and kurtosis of that variable's values and testing those statistics against a False Discovery Rate criterion. This approach was time-consuming and sensitive to outliers. In addition, the transformation itself was relatively *ad hoc*.

We now base our decision to transform on robust measures of skewness and kurtosis called L-skewness and L-kurtosis [Hosking 1990]. L-moments are based on order statistics and are robust against outliers and generalize to a larger family of distributions than do ordinary power-moments. In addition, computing L-moments requires only a sort plus a pass through half the values.

A second important modification of our original transformation strategy is the separation of transforms to alleviate skewness and kurtosis. If the absolute value of L-skewness exceeds .2, we apply the transformation:

$$y^* = log(y/(1-y))/20$$

Otherwise, if L-kurtosis exceeds .2 (a peaked or *leptokurtic* distribution), we apply the transformation:

$$y^* = 1/(1+exp(-6y+3))$$

Figure 2 shows these two transforms. Our skewness transform resembles the well-known logit function. Our kurtosis transform resembles the familiar logistic sigmoid function. The point distribution above each plot shows the effect of the rescaling on an equally-spaced set of points. For the skewness transform, the outer regions of the unit interval are magnified and for the kurtosis transform, the middle section of the interval is magnified.

Our goal in transforming is to improve our chances of discovering class separation in relatively dense regions. This is especially important because we use binning (a form of segmentation) to compress our data. Without transformation, highly-skewed or kurtotic densities might concentrate in only one or two bins. The logic behind this is similar to the rationale for using linearizing transformations in support vector machines.

The next three subsections describe the three stages that comprise the core of the CHIRP algorithm. We iterate these three stages cyclically over classes until we are unable to classify remaining training data. Each iteration is a one-against-all classification step involving the current class vs. other classes.

3.2. Projecting

We compute 2D projections of variables in the hope of locating dense and well-separated class distributions. To do this, we generate a candidate list of 1D projections, pick the best of these based on a separation measure for the current class, and pair the best to make a set of 2D projections.

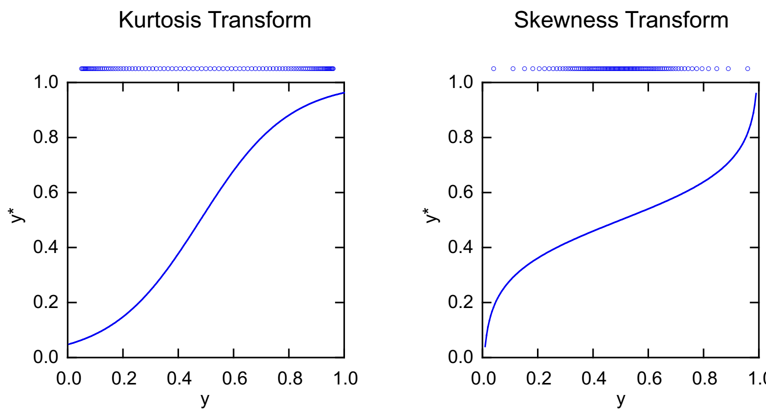

Fig. 2. Kurtosis and Skewness Transforms.

Before projecting, we need to scale categorical variables in order to project them into the same subspace along with continuous variables. To scale categorical variables, we use a strategy derived from the latent class model [Lazarsfeld and Henry 1968]. For a given categorical variable, we count the unclassified instances of the current class in each category. We divide this count by the total count of unclassified instances in each category. Finally, we replace integer category values with the corresponding proportions based on these two counts.

Next, we generate a set of 1D projections using three-valued vectors with elements

$$u_j \in \{-1, 0, 1\}, j = 1, \ldots, p.$$

Of the p projection weights, r are zero and the remainder are split evenly between -1 and 1.

Choosing r depends on p. When p is small ($p \leq 50$), we apply random projections with zero and nonzero weights. Otherwise, we apply random projections after constraining $p - 50$ weights to be zero. Our choice of 50 is guided by results in [Hegde and Baraniuk 2007] and [Li et al. 2006].

3.2.1. Small p. If p is small, we choose $r = p/4$, $r = p/2$, or $r = 3p/4$. We decide among the three alternatives by generating three random projections (using these r values) and choosing the one with the largest value of a separation statistic S. For a projection, our separation statistic is the distance of the current-class projected mean $\bar{x}_c$ from the closest other-class projected mean $\bar{x}_k$:

$$S = \min_{k \neq c}(d_{\bar{x}_c, \bar{x}_k})$$

3.2.2. Large p. If p is large, we set $p - 50$ weights to zero before doing our random projections on the remaining features the same way we do for small p. In this case, we need to determine which features are constrained to have zero weights. To decide, we compute the class-separation statistic S on each variable. We sort all features on this statistic and we constrain the $p - 50$ features with the smallest class-separation statistics to have zero weights. This process is a form of feature selection, but unlike other applications that use feature selection to pre-process large datasets, we employ it *inside* our iterations. Different features are likely to be selected on different iterations because class means change as points are removed from the training set.

Unit-weighting our projections is a form of regularization [Hastie et al. 2001; Tibshirani 1995]. Regularized estimators increase bias in order to reduce prediction error. We discuss this aspect further in the Appendix.

3.3. Binning

The next step in the process is to pair our best 1D projections and bin currently unclassified instances into a bin matrix for each pair. We base the number of bins for each 2D projection on a formula in [Sturges 1926]. Given n instances, we compute the marginal number of bins b using

$$b = 2\ log_2(n)$$

This formula produces a few more bins than optimal statistical estimates for binning normal and mildly skewed distributions [Scott 1979; Wand 1997]. Traditional methods assume a homogeneous distribution, however, which is clearly not the case in classification.

Next, we rank our $b \times b$ 2D bin matrices on a purity measure. For a given target class C_k, our purity measure is

$$P_k = \sum_{i=1}^{b} \sum_{j=1}^{b} n_{i,j} I_{i,j}(C_k)$$

where

$$I_{i,j}(C_k) = \begin{cases} 1 & n_{i,j} = n_{i,j,k} \\ 0 & otherwise \end{cases}$$

In other words, we sum the counts across all bins whose total counts of points falling in them ($n_{i,j}$) are due only to class C_k counts ($n_{i,j,k}$). We want our purity measure to count only pure bins, because our fitting method will be especially greedy. The more pure bins we can eliminate early in the process, the better chance we have of seeing well-separated other classes later.

To recapitulate our current situation: we have generated a small number of 1D random projections sorted on our separation measure S and we have paired them to make a set of 2D projections. We have then chosen the best of these 2D projections based on our bin purity measure P. We are now working with the upper tail of the extreme-value distribution of binned, unit-weighted random projections ordered on a bin purity measure. We now will cover these binned projections with rectangles and pick the cover that most improves our training-set classification.

3.4. Covering

The last stage in each iteration involves covering pure bins in order to define a classification region for a given class C_k. Our cover is a CHDR, which is a list of HDRs. Each CHDR is uniquely associated with a class label.

3.4.1. Growing a CHDR. We developed a covering algorithm after observing humans select homogeneous regions in a classification "game" [Anand et al. 2009]. Figure 3 shows how this process works. For a given pure bin element $b_{i,j}$, we grow an HDR covering the bin and its pure neighbors by expanding upward ($b_{\cdot,j+1}$), rightward ($b_{i+1,\cdot}$), downward ($b_{\cdot,j-1}$), and leftward ($b_{i-1,\cdot}$) in a spiral path. In other words, we sequentially expand each side of the current rectangle by one bin-row or bin-column whose length is equal to the length of that side. We cease expanding in any of the four directions when the odds ratio of current-class vs. other-class instances inside the covering rectangle

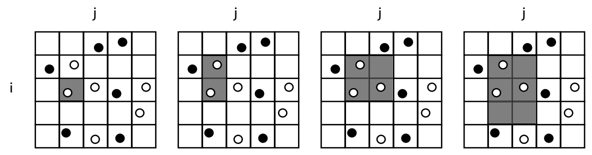

Fig. 3. Growing a Hypercube Description Region (HDR) on a binned 2D projection. Each point is located at the centroid of the instances in each cell. Hollow symbols represent bins containing only instances of the current class. Solid symbols represent bins containing at least one instance of another class.

begins to decrease. This strategy tends to result in squarish rectangles that cover pure or empty bins, similar to an approach in [Agrawal et al. 1998].

We grow an HDR for each of the bins in the 2D bin matrix. For each HDR we record the number of instances of the current class that we have covered. We pick the HDR that results in the largest current-class count. Finally, if the current-class count in the HDR exceeds 10, we add the HDR to the current CHDR list for that 2D projection.

This 10 is not a magic number. It is based on a rule-of-thumb for a slippage test. [Tukey 1959] wrote:

> Given two groups of measurements, taken under conditions (treatments, etc.) A and B, we feel the more confident of our identification of the direction of difference the less the groups overlap one another. If one group contains the highest value and the other the lowest value, then we may choose (i) to count the number of values in the one group exceeding all values in the other, (ii) to count the number of values in the other group falling below all those in the one, and (iii) to sum these two counts (we require that neither count be zero). If the two groups are of roughly the same size, then the critical values of the total count are, roughly, 7, 10 and 13, i.e. 7 for a two sided 5% level, 10 for a two sided 1% level, and 13 for a two sided 0.1% level.

Our application fits this description because we construct an HDR to cover only instances outside the range of other-class instances. There are some caveats, of course. Our count of other-class instances is often substantially greater than the count of current-class instances inside an HDR; Tukey's approach assumes relatively balanced sample sizes. Second, we work in 2D; Tukey worked in 1D. Third, we count only highest values; Tukey counted highest and lowest. Tukey discusses several adjustments to deal with these problems, but we found little need to employ them since our method biases the test in a conservative direction. See [Mosteller 1948] for more information on slippage tests.

Once we compute an HDR, we mark bins that it covers. Then we iterate this procedure over the 2D bin matrix starting with uncovered bins until we can find no HDRs that meet Tukey's criterion. The resulting set of HDRs is a CHDR for a 2D bin matrix.

3.5. Iterating

We iterate through classes in cyclical order. For each iteration, we pick a new target class and repeat our three stages (*projecting*, *binning*, *covering*). This means recalculating all the statistics within these stages. Fortunately, these are one-pass calculations, so the iterations are fairly rapid. Furthermore, later iterations are faster than early ones because peeling away classified points results in fewer points to bin and test. We terminate iterations when no CHDR can be constructed according to our rules.

3.6. Missing Values

We use weighted projections for data vectors with missing values. This involves computing a linear combination with weights applied to nonmissing values. The result is then adjusted by p/p^-, where p is the number of weights and p^- is the (smaller) number of nonmissing values. This is a rather poor imputation method. Much better would be a nearest-neighbor or maximum-likelihood estimate, although we found these too expensive for our applications.

4. CHIRP SCORING

To score, we normalize and transform a new point. Then we pass through the list of CHDRs. For each CHDR, we project the point using the stored projections from the training data. Then we pass through the list of rectangles for that CHDR. The first rectangle to enclose our projected testing point determines the classification.

If no enclosing rectangle is encountered by the end of the list, we assign the point to the nearest rectangle in the CHDR list. This computation involves finding the shortest L^∞ distance between a point and a rectangle. Because the perimeter of a CHDR is a zero level set for a naive density estimator based on the union of rectangular polygons [Silverman 1986], this point-to-rectangle distance is asymptotically a nearest-neighbor statistic.

This scoring algorithm is based on a decision list [Rivest 1987]. Unlike trees, decision lists do not require traversal of the entire depth in order to score new instances (unless, of course, a cover is not encountered).

5. PERFORMANCE

In this section we will present performance statistics for CHIRP. Because we have substantially improved the performance of CHIRP since the original KDD paper, we have recomputed the CHIRP statistics. We will discuss two different aspects of CHIRP performance: accuracy and efficiency. We conducted an experiment to evaluate CHIRP against a comprehensive set of competitive classifiers. We first summarize the experimental design and then present results for accuracy and efficiency.

5.1. Datasets

We tested CHIRP and other classifiers on 20 datasets from the UCI Machine Learning Repository [Asuncion and Newman 2007], [Hastie et al. 2001], and other sources. Table I summarizes prominent aspects of these datasets.

5.2. Challenges when Evaluating Classifiers

There are at least three reasonable questions for proponents of particular classifiers who conduct evaluation experiments: 1) Have they “cherry-picked” their datasets to make their classifiers look effective? 2) Have they included a sufficient number of datasets to provide reasonable statistical power for their conclusions? and 3) Have they tested their classifiers against a sufficient number of competitors to insure their claims are generalizable?

In response to the first question, we selected these particular datasets for their structural variety; each represents a different challenge for classifiers. We tried not to bias the results by picking two or more datasets with a similar structure. These datasets include examples of missing values (Horse), mixed categorical and continuous variables (Adult), mixed binary variables (Cover), small n (Spect), large n (Poker, Adult), small p (Swiss Roll), large p (Madelon, Cancer), $n \ll p$ (Cancer), small g (Adult, Credit, Horse, ...), large g (Cancer), and relatively small ratios of training to testing instances (Poker, Segment). We also looked for datasets with disparate within-and-between-groups data

Table I. Characteristics of Datasets

	Training	Testing	Attributes	Groups	Categorical Vars	Continuous Vars
Abalone	2,088	2,089	8	3	No	Yes
Adult	32,561	16,281	14	2	Yes	Yes
Cancer	144	54	16,063	14	No	Yes
Cover	11,340	569,672	54	7	Yes	Yes
Credit	345	345	14	2	No	Yes
Horse	300	68	22	2	Yes	Yes
Madelon	2,000	600	500	2	No	Yes
Optdigits	3,823	1,797	64	10	Yes	Yes
Orange10	5,000	50,000	10	2	No	Yes
Page Blocks	4,000	1,473	10	5	No	No
Pendigits	7,494	3,498	16	10	No	Yes
Poker	25,010	1,000,000	10	10	No	Yes
Satellite	4,435	2,000	36	6	No	Yes
Segment	210	2,100	19	7	No	Yes
Shuttle	43,500	14,500	9	7	No	Yes
Spect	80	187	22	2	Yes	No
Swiss Roll	1,000	1,000	3	2	No	Yes
Vehicle	679	167	18	4	No	Yes
Vowel	528	462	10	11	No	Yes
Waveform	300	500	21	3	No	Yes

densities (discrete, continuous, mixed, convex, non-convex, etc.). Almost all the test datasets are real, and each has been widely tested on numerous classifiers.

In response to the second question, we included 20 datasets. This number enabled relatively narrow confidence intervals on our error results. The median width of our confidence intervals for standardized errors was .6. The distributions of the standardized errors within classifiers are relatively symmetrical, so our use of the t-distribution to construct confidence intervals is justified..

In response to the third question, we added CHIRP to the Weka data mining workbench [Witten et al. 1999]. Then we tested every classifier in Weka Version 3.6.1, omitting classifiers that could not deal with all 20 datasets (because they were specialized or were not scalable to the larger datasets). This left a total of 50 classifiers. We included hybrid and meta classifiers as well, even though these are not direct competitors because they do not rest on a single geometric model. Tests were run on a 2.5 GHz Intel Core 2 Duo Macintosh Powerbook with Macintosh OS X Version 10.5.7 and Java Version 1.5.0 running in a 2GB partition. The full experiment took almost three weeks of continuous CPU time.

To the best of our knowledge, this is one of the most comprehensive experimental evaluations of classifiers since the Statlog Project [King et al. 1995; Statnikov et al. 2005; Abdullah et al. 2006]. We also examined published error rates for non-Weka classifiers on these datasets and found almost all of them to lie in the range of our findings (except for specialized classifiers such as [Li 2010], which can perform extraordinarily well on specific types of datasets).

5.3. Accuracy

We computed a variance components analysis on the error rates for every classifier across datasets. Dataset and Classifier were treated as random factors. The effects of both factors were highly significant ($p < .001$). Consequently, we standardized the error statistics within dataset for our final analysis.

Figure 4 summarizes the error performance for all classifiers. CHIRP has the lowest standardized error of all classifiers and a relatively small variance. CHIRP is not only extraordinarily accurate but also unusually consistent across a wide range of data scenarios.

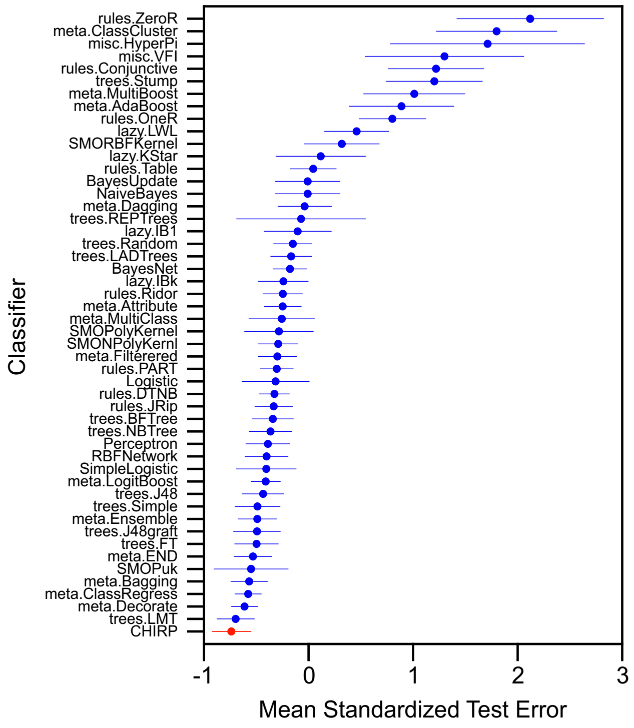

Fig. 4. Average standardized error rates and associated 95% confidence intervals for CHIRP and Weka Version 3.6.1 classifiers. Default parameter values were used for all classifiers.

Figure 5 shows the performance of all the classifiers on these datasets. CHIRP is highlighted in red. No other family of classifiers had the lowest or near-lowest error on as many datasets as did CHIRP. There are several remarkable findings in this plot. First, CHIRP substantially beats the other classifiers on the Cancer dataset. The simple feature-selection algorithm using the separation index appears effective. This strategy resembles the common practice of computing t-statistics on genes in microarray research in order to do model selection. Second, CHIRP excels on the Poker Hand dataset; the other classifiers handle it poorly. Ping Li recommended this dataset because his boosting classifiers, designed specifically for this type of data, have achieved over 90% accuracy on Poker. We were surprised to discover how well CHIRP did here, since we did not consider the peculiar aspects of that dataset in designing CHIRP. Third, CHIRP achieves the lowest error for the Swiss Roll dataset. It is able to find enough revealing projections to peel away and extract class information from this difficult nonlinear manifold.

5.4. Efficiency

Figure 6 shows the training times for the new CHIRP and the other classifiers. Not surprisingly, some of the worst performing classifiers in Figure 4 are the fastest to train. The converse is not always true, however. The Multilayer Perceptron classifier was the least efficient to train, yet its performance was middling. CHIRP's closest rival in accuracy, Logistic Model Trees, was substantially slower in training.

The new CHIRP is more than three times faster than the original. Much of this improvement came from attention to memory usage and adaptations to the Weka architecture. Nevertheless, the new CHIRP is still slower to train than some of the other

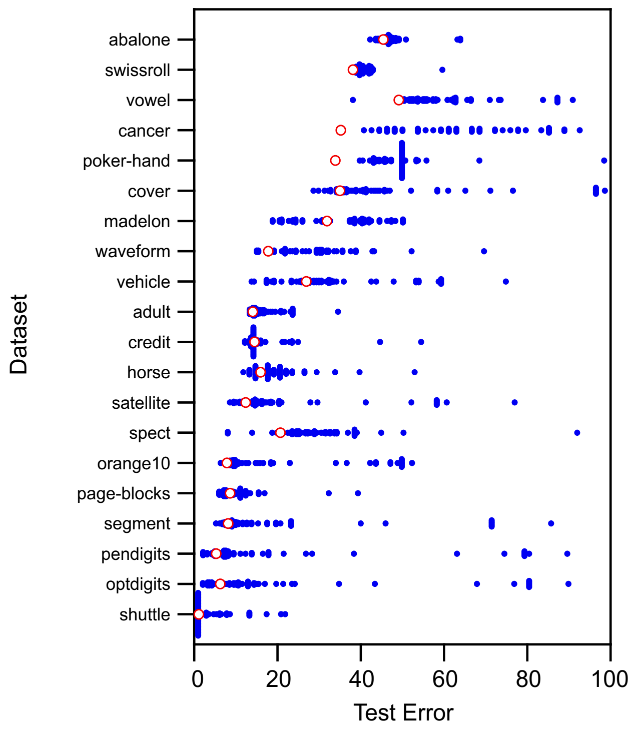

Fig. 5. Errors for CHIRP (in red) and the other classifiers (in blue) on 20 test datasets.

classifiers. In the main, this is due to its ensemble architecture. In a parallel computing environment, each thread would run concurrently. This would be trivial to implement on a multi-core processor or in a parallel environment such as Map Reduce. As with Random Forests, it would take minimal algorithmic modifications to run CHIRP in separate threads. Even without these modifications, however, CHIRP is reasonably efficient. The longest training time for other classifiers (rulesDTNB on Madelon) was 38 hours. The longest training time for CHIRP was 27 minutes on Poker Hand. The longest time for an SVM was 15 hours (SMOPuk on Adult).

Figure 7 shows the testing times per instance for the new CHIRP and the other classifiers. In contrast to its training performance, CHIRP falls in the fastest group of classifiers, with performance that is not significantly worse than decision trees (although its standard error is somewhat larger). Interestingly, the support vector machines are generally the slowest in testing; the SVM with the lowest overall error (SMOPuk) is among the slowest performers in a testing environment.

Again, scoring could be speeded up for CHIRP by parallelizing the voting. The longest scoring time for any classifier on one instance was 4 minutes (lazyKStar on Satellite). The longest scoring time for CHIRP was less than a tenth of a second (on Satellite). The longest scoring time for an SVM was 22 seconds (SMOPuk on Satellite). These longer times are problematic, because scoring times of more than a few seconds would be impractical for online applications. By contrast, CHIRP is a good candidate for online classification in a time-critical environment.

5.4.1. Theoretical Performance. CHIRP makes one pass through n rows of the training data to compute data limits and basic statistics. For each of the g classes, it makes an additional pass through the data to construct 25 2D bin matrices. CHIRP sorts this

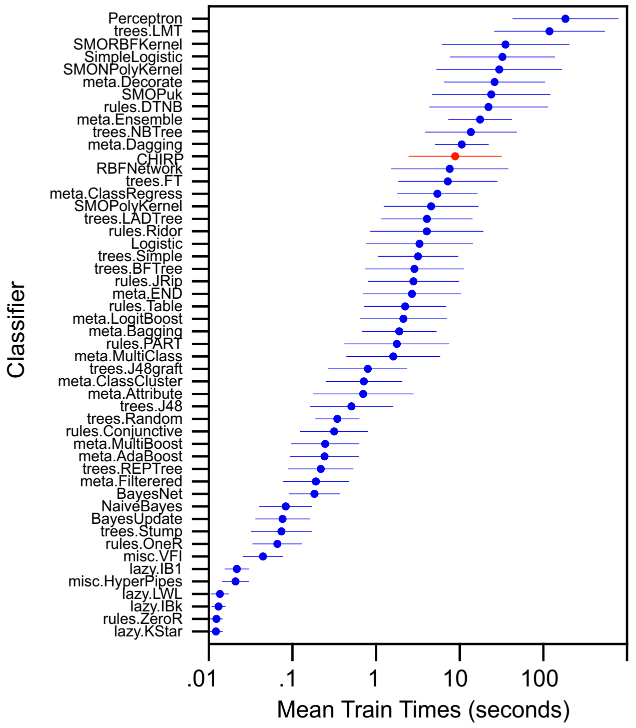

Fig. 6. Mean training times and associated 95% confidence intervals for CHIRP and Weka Version 3.6.1 classifiers.

bin-matrix list and picks the top 5 candidate 2D bin matrices. It iterates through this process m times, adding a CHDR to the decision list at each step. Thus, we should expect CHIRP to be $O(gmnp)$ in time. To test this expectation, we did a simulation.

We generated spherical Gaussians for $n = \{500, 5000, 50000\}$, $p = \{20, 40, 60, 80, 100\}$, and $g = \{2, 4, 6, 8, 10\}$. In each of the 75 datasets, the first g Gaussians had unit variance with centroids located at the corners of a $(g-1)$-simplex with edges of length 7. Values for the remaining $p - g$ variates were $N(\mathbf{0}, \mathbf{I})$.

Figure 8 shows a graph of the performance of CHIRP on these random datasets. We have enhanced the plot with a distance-weighted least-squares smoother. The points are fit well ($R = .96$, with well-behaved residuals) by the simple linearized model:

$$E[\log(t)] = -11.389 + 0.942 \log(n) + 1.655 \log(g) + 0.677 \log(p)$$

Our empirical results show that CHIRP is sub-linear in n and p and super-linear in g. It would appear that CHIRP is not the best candidate for problems involving hundreds or thousands of classes. For relatively small g, however, CHIRP performance is similar to that of k-means clustering, which is $O(gnp)$ on g clusters, n cases, and p variables.

6. DISCUSSION

Our experiment provides clear evidence that CHIRP outperforms competitive classifiers across a wide range of well-known datasets used to evaluate classifiers. We intend to investigate in more detail how this happened, although we designed each stage using well-established findings from the statistical and machine-learning literature. What is unique about CHIRP is how these components are assembled. We need to investigate how these components interact.

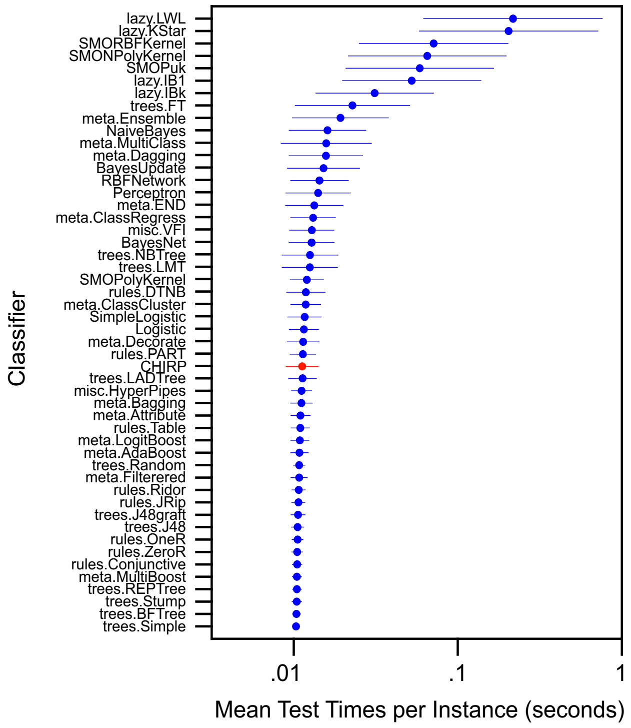

Fig. 7. Mean testing times per instance and associated 95% confidence intervals for CHIRP and Weka Version 3.6.1 classifiers.

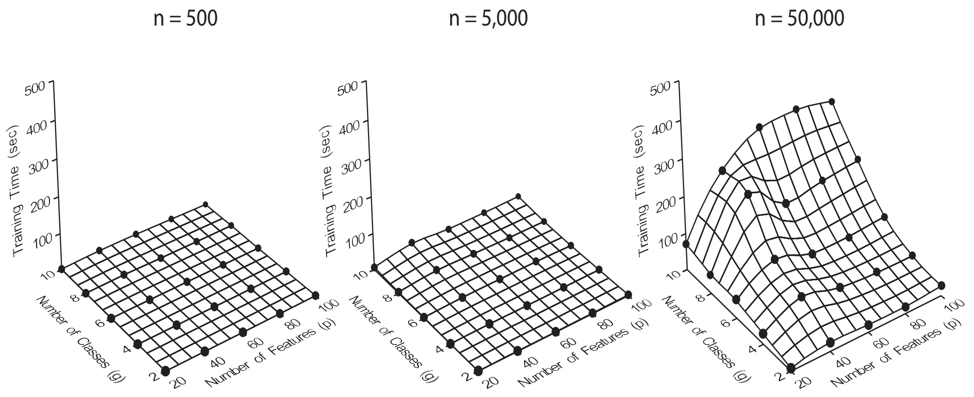

Fig. 8. Training times for CHIRP on random datasets. The parameters in these plots are number of classes (g), number of features (p), and number of instances (n).

We do have several preliminary answers to the question of why CHIRP works so well. First, CHIRP handles nonconvex, discrete and disjoint densities by covering instead of partitioning. We suspect (but have not yet proven) that covering requires fewer rectangles than partitioning in the case of certain topologies (such as the Orange10 or Waveform datasets). Second, its categorical scaling algorithm allows us to combine discrete and continuous densities to search for homogeneous joint regions. Third, CHIRP is an ensemble classifier. Its use of random projections naturally lends itself to a voting architecture. Fourth, CHIRP uses affine instead of axis-parallel projections; other set-covering classifiers do not employ this technique. Fifth, the two fitness measures used for ranking projections (the class separation measure S and the bin purity measure P) target different aspects of densities. The S measure values projections with large margins; the P measure values projections with compact subsets. Projections missed by one are likely to be found by the other. Finally, CHIRP embeds its projections and covers *inside* its iterations; we have, for the first time, used projections and covers recursively. This architecture contributes to the ability of CHIRP to peel away sets of exterior points that obscure other sets in the core of a density. This peeling is adaptive; CHIRP responds to the topology of the conditional density as it shrinks in size and changes shape on each iteration.

CHIRP can have difficulty with certain higher-dimensional densities because its covers are confined to 2D projections. Its ability to peel off subsections of higher-dimensional densities mitigates against this weakness, but there are some configurations it cannot exploit. Like all classifiers, CHIRP cannot outperform everyone on every dataset.

There were several questions raised about CHIRP during and after the KDD discussion session. Some of these deserve notice. One questioner recommended that we use the ROC area under curve (AUC) instead of average error as a criterion for comparing classifiers. Setting aside the well-known problems with AUC [Hanczar et al. 2010; Hand 2010], we must point out that CHIRP does not provide any sort of classification probability estimate that would be needed for computing AUC. We have explored this possibility through interpreting the CHDRs as fixed rectangular kernels, but this effort has so far led nowhere. In any case, we do not believe the results would differ substantially, since AUC and average error are highly correlated when marginal distributions are relatively balanced, as is the case with most of our datasets.

One commented that we should have used our categorical coding algorithm to preprocess categorical variables for other classifiers. This would be impossible, since CHIRP does categorical recodes on every iteration. It is an essential and non-separable part of our algorithm. We think it is a distinct advantage of our design.

Finally, a Weka team member mentioned that the default tree-number setting for Random Forests was a poor choice. He recommended changing this number to 100. The change made a huge difference. Random Forests moved from thirty-second to third place, behind Logistic Model Trees. The size of this improvement raises a question about the settings for other classifiers. Since our simulation requires almost a month of CPU time, customizing settings to improve the performance of individual classifiers is not practical. We expect supporters of specific classifiers will fiddle with parameter values in order to improve their performance on these datasets and we welcome that effort. We must repeat, however, that customizing settings for individual datasets defeats the purpose of any comparison. In plain language, we believe that the rules for our experiment take some of the “black art” out of classifier comparisons. If default settings can be improved (as Weka will be doing with Random Forests in the next release), then they should become defaults for *all* datasets and the Weka team should be responsible for making the change. Because most published classifier comparisons have been idiosyncratic, opportunistic, and selective, their variability precludes objective evaluations. To deal with this pervasive problem, we recommend that published evaluations of new classifiers or modifications to existing ones be conducted in Weka on a suite of test datasets that includes at least the ones we have employed. This procedure would provide objective historical statistics as well as meaningful timing data.

7. CONCLUSION

Producing an error rate that matches or exceeds other known classifiers on a wide range of datasets is not the principal distinction of this research. Such a result could be accomplished through an incremental improvement of any of the most competitive classifiers. Indeed, this is what sets the bar so high for a new classifier; the three or four leading classifier frameworks have been polished for decades.

Accomplishing extraordinary accuracy with a novel classifier is what distinguishes this research. The CHIRP project began with a simple idea: to link a visually-motivated covering algorithm to a random projection machine. It evolved with the realization that random projections could be nested within iterations. And it concluded with the discovery that covers on random projections could be used to peel away subsets of points to unwrap high-dimensional nonlinear configurations. Because this architecture is so new, we expect to see substantial improvements in the future.

The result so far:

— The performance is sub-quadratic in complexity on n, p, and g (number of instances, number of features, and number of classes).
— CHIRP does not depend on sensitive adjustable parameters (convergence criteria, kernel types, bandwidths, pruning schedules, etc.). We tested this assertion by assessing its performance over a wide range of parameter settings. Most importantly, none of the potentially settable CHIRP parameters was adjusted to optimize performance on a specific dataset in our training or testing.
— CHIRP had the lowest average standardized testing error rate and achieved the lowest error rate on more datasets than did any other classifier. Clearly, these datasets were not selected to favor CHIRP; we included well-known datasets designed to present classifiers with the broadest variety of challenges.
— CHIRP is readily parallelizable at the random projection stage and/or voting stage.
— CHIRP is simple and tiny. Its JAR file is under 50K in size. The algorithm iterates over only three steps.
— CHIRP is a novel algorithm; it is not a hybrid classifier. This fact would tend to support the idea that CHIRP can contribute relatively independent classification information to the results of other classifiers.

Given these distinctive features and its fundamental differences from other classifiers, CHIRP is uniquely suited for applications where there is limited or no *a priori* knowledge of the process generating the data.

REFERENCES

ABDULLAH, M. R., TOH, K.-A., AND SRINIVASAN, D. 2006. A framework for empirical classifiers comparison. In *Industrial Electronics and Applications*. IEEE.

ACHLIOPTAS, D. 2001. Database-friendly random projections. In *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, New York, 274–281.

AGRAWAL, R., GEHRKE, J., GUNOPULOS, D., AND RAGHAVAN, P. 1998. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD*. 94–105.

AGUILAR, J., RIQUELME, J., AND TORO, M. 1998. Decision queue classifier for supervised learning using rotated hyperboxes. In *Proceedings of the 6th Ibero-American Conference on AI: Progress in Artificial Intelligence*. Lecture Notes in Computer Science Series, vol. 4045. Springer, 326–336.

ALPERN, B. AND CARTER, L. 1991. The hyperbox. In *Proceedings of the IEEE Information Visualization 1991*. 133–134.

ANAND, A., WILKINSON, L., AND TUAN, D. N. 2009. An L-infinity norm visual classifier. In *ICDM*. 687–692.

ASUNCION, A. AND NEWMAN, D. 2007. UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

BICKEL, P. AND LEVINA, E. 2004. Some theory for Fisher's linear discriminant function, 'naive Bayes', and some alternatives when there are many more variables than observations. *Bernoulli 10*, 989–1010.

BREIMAN, L., FRIEDMAN, J., OLSHEN, R., AND STONE, C. 1984. *Classification and Regression Trees*. Wadsworth, Belmont, CA.

BU, S., LAKSHMANAN, L. V. S., AND NG, R. T. 2005. MDL summarization with holes. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 433–444.

DIETTERICH, T. G. AND BAKIRI, G. 1995. Solving multiclass learning problems via error-correcting output codes. *Journal of AI Research 2*, 263–286.

FLICK, T. E., JONES, L. K., PRIEST, R. G., AND HERMAN, C. 1990. Pattern classification using projection pursuit. *Pattern Recognition 23*, 1367–1376.

GAO, B. 2002. Hyper-rectangle-based discriminative data generalization and applications in data. Ph.D. thesis, Simon Fraser University.

GAO, B. J. AND ESTER, M. 2006. Turning clusters into patterns: Rectangle-based discriminative data description. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*. IEEE Computer Society, Washington, DC, USA, 200–211.

GUO, Y., HASTIE, T., AND TIBSHIRANI, R. 2005. Regularized discriminant analysis and its application in microarrays. *Biostatistics 1*, 1–18.

HANCZAR, B., HUA, J., SIMA, C., WEINSTEIN, J., BITTNER, M., AND DOUGHERTY, E. R. 2010. Small-sample precision of roc-related estimates. *Bioinformatics 26*, 822–820.

HAND, D. 2010. Measuring classifier performance: A coherent alternative to the area under the roc curve. *Machine Learning 77*, 103–123.

HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. H. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York.

HEGDE, C. AND BARANIUK, R. 2007. Random projections for manifold learning. In *NIPS 2007: Proceedings of the 2007 conference on Advances in neural information processing systems*. MIT Press, Cambridge, MA, USA.

HOSKING, J. 1990. L-moments: analysis and estimation of distributions using linear combinations of order statistics. *Journal of the Royal Statistical Society,* Series B *52*, 105–124.

JIMENEZ, L. O. AND LANDGREBE, D. A. 1995. Projection pursuit for high dimensional feature reduction: paralleland sequential approaches. In *Geoscience and Remote Sensing Symposium, 1995. IGARSS '95*. Vol. 1. 148–150.

JOHNSON, W. B. AND LINDENSTRAUSS, J. 1984. Lipschitz mapping into Hilbert space. *Contemporary Mathematics 26*, 189–206.

KING, R., FENG, C., AND SUTHERLAND, A. 1995. Statlog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence 9*.

LAZARSFELD, P. F. AND HENRY, N. 1968. *Latent Structure Analysis*. Houghton Mifflin, Boston.

LEE, E.-K., COOK, D., KLINKE, S., AND LUMLEY, T. 2005. Projection pursuit for exploratory supervised classification. *Journal of Computational and Graphical Statistics 14*, 831–846.

LI, P. 2010. Robust LogitBoost and Adaptive Base Class (ABC) LogitBoost. In *UAI 2010 Proceedings*. IEEE.

LI, P., HASTIE, T. J., AND CHURCH, K. W. 2006. Very sparse random projections. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 287–296.

MARCHAND, M. AND SHAWE-TAYLOR, J. 2002. The set covering machine. *Journal of Machine Learning Research 3*, 723–746.

MOSTELLER, F. 1948. A k-sample slippage test for an extreme population. *The Annals of Mathematical Statistics 19*, 58–65.

PU, K. Q. AND MENDELZON, A. O. 2005. Concise descriptions of subsets of structured sets. *ACM Transactions on Database Systems 30,* 1, 211–248.

QUINLAN, J. R. 1993. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann.

RIVEST, R. L. 1987. Learning decision lists. *Machine Learning 2*, 229–246.

SCOTT, D. W. 1979. On optimal and data-based histograms. *Biometrika 66*, 605–610.

SILVERMAN, B. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, New York.

SIMPSON, P. K. 1992. Fuzzy min-max neural network, i: Classification. *IEEE Transactions on Neural Networks 3*, 776–786.

SOKOLOVA, M., JAPKOWICZ, N., MARCHAND, M., AND SHAWE-TAYLOR, J. 2003. The decision list machine. In *Advances in Neural Information Processing Systems 15*. MIT Press, 921–928.

STATNIKOV, A., ALIFERIS, C. F., TSAMARDINOS, I., HARDIN, D., AND LEVY, S. 2005. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics 21,* 5, 631–643.

STURGES, H. A. 1926. The choice of a class interval. *Journal of the American Statistical Association 21*, 65–66.

TIBSHIRANI, R. 1995. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society,* Series B *58*, 267–288.

TUKEY, J. 1959. A quick, compact, two-sample test to Duckworth's specifications. *Technometrics*, 31–48.

ÜNEY, F. AND TÜRKAY, M. 2006. A mixed-integer programming approach to multi-class data classification problem. *European Journal of Operational Research 173*, 910–920.

WAINER, H. 1976. Estimating coefficients in linear models: It don't make no nevermind. *Psychological Bulletin 83,* 2, 213–217.

WAND, M. P. 1997. Data-based choice of histogram bin width. *The American Statistician 51,* 1, 59–64.

WILKINSON, L., ANAND, A., AND DANG, T. 2011. CHIRP: A new classier based on composite hypercubes on iterated random projections. In *Proc. of ACM Conf. on Knowledge Discovery and Data mining*.

WITTEN, I. H., FRANK, E., TRIGG, L., HALL, M., HOLMES, G., AND CUNNINGHAM, S. J. 1999. Weka: Practical machine learning tools and techniques with Java implementations. In *Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop on Emerging Knowledge Engineering and Connectionist-Based Information Systems*. 192–196.

8. APPENDIX

Suppose there are two normal populations with respective $p \times 1$ mean vectors $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ and common $p \times p$ covariance matrix $\mathbf{\Sigma}$. Without loss of generality, we will assume that $\boldsymbol{\mu_1} = -\boldsymbol{\mu_2}$. The two-group Linear Discriminant Analysis (LDA) classification algorithm assigns a new point x to the group with the smaller Mahalanobis distance

$$(\mathbf{x} - \boldsymbol{\mu}_i)^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_i), i = 1, \ldots, 2$$

Equivalently, if the Fisher linear discriminant function

$$\delta_F(\mathbf{x}) = \mathbf{\Sigma}^{-1} (\boldsymbol{\mu_1} - \boldsymbol{\mu_2})^T \mathbf{x}$$

is negative, we assign x to the first group, otherwise to the second.

Because we do not usually know $\boldsymbol{\mu_1}$ or $\boldsymbol{\mu_2}$ or $\mathbf{\Sigma}$, we customarily estimate them via maximum likelihood on n observations of sample data and employ the discriminant function

$$d_F(\mathbf{x}) = \widehat{\mathbf{\Sigma}}^{-1} (\hat{\boldsymbol{\mu}}_\mathbf{1} - \hat{\boldsymbol{\mu}}_\mathbf{2})^T \mathbf{x}$$

for our classification rule.

When $p > n$, the maximum likelihood estimate of $\mathbf{\Sigma}$ cannot be computed because the conventional matrix estimator is singular. Classical remedies for computing the linear discriminant function in these cases include using a Moore-Penrose inverse or selecting a subset of the p variables (features) to get our estimate.

Alternatively, we can assume $\mathbf{\Sigma} = \sigma^2 \mathbf{I}$. In this case, the estimated discriminant function passes through $\hat{\boldsymbol{\mu}}_\mathbf{1}$ and $\hat{\boldsymbol{\mu}}_\mathbf{2}$ and the decision rule based on the linear discriminant function is equivalent to a Naive Bayes rule. Bickel and Levina [Bickel and Levina 2004] prove that the Naive Bayes classification rule substantially outperforms the Fisher linear discriminant rule under broad conditions when the number of variables grows faster than the number of observations. This gives us some confidence that we do not substantially increase prediction error by ignoring covariance structure when searching for maximum separation of means in higher-dimensional spaces.

Suppose we now replace the discriminant function coefficients with unit weights

$$d_U(\mathbf{x}) = \mathbf{u}^T \mathbf{x},$$

where $u_i \in \{-1, 0, 1\}$. Suppose also that we choose unit weights that produce the greatest spread between sample means on the $d_U(\mathbf{x})$ discriminant function. The number of possible weights we must consider before making this choice is $\frac{1}{2}(3^p - 1)$. (The $\frac{1}{2}$ is due to the symmetry of $d_U(\mathbf{x})$ around zero).

The following lemma gives us the upper bound of the angle between the optimal unit-weight vector $d_U(\mathbf{x})$ and the Fisher discriminant vector $d_F(\mathbf{x})$.

LEMMA 8.1. *Let U be the set of all non-null $p \times 1$ vectors $\mathbf{u}$, where $u_i \in \{-1, 0, 1\}$. Let $\mathbf{x}$ be a $p \times 1$ vector in $\boldsymbol{R}^p$. Let $\mathbf{u_x}$ be the element of U that is closest in angle to $\mathbf{x}$. Then for any $\mathbf{x}$, the maximum possible angle between $\mathbf{u_x}$ and $\mathbf{x}$ is*

$$\theta_{max} = \arccos\left(1/\sqrt{p^2 - 2\sum_{m=1}^{p} \sqrt{m}\sqrt{m-1}}\right)$$

For $p = 50$, for example, using $d_U(\mathbf{x})$ instead of $d_F(\mathbf{x})$ will shrink the values of $\hat{\boldsymbol{\mu}}_1$ and $\hat{\boldsymbol{\mu}}_2$ projected on $d_F(\mathbf{x})$ toward zero by a factor of approximately .3. The gains from this type of shrinkage are discussed in [Wainer 1976; Hastie et al. 2001; Tibshirani 1995; Guo et al. 2005] and elsewhere. It belongs to a class of regularization methods that, relative to maximum likelihood estimators like $d_F(\mathbf{x})$, are more resistant to outliers and have lower prediction error in new samples.

In practice, we cannot expect to find the projection $d_U(\mathbf{x})$ with the greatest separation of means because it is impractical to search over $\frac{1}{2}(3^p - 1)$ weight vectors for large p. We can get close, however, by taking advantage of the Johnson-Lindenstrauss Theorem [Johnson and Lindenstrauss 1984]. This theorem states that if a metric on X results from an embedding of X into a Euclidean space, then X can be embedded in R^k with distortion less than $1 + \epsilon$, where $k = O(\epsilon^2 log|X|)$. Remarkably, this embedding is achieved by projecting onto a random k-dimensional subspace. Because our discriminant rule depends on a similarity transformation of Euclidean distances, we can logarithmically reduce the complexity of the problem through random projections.

Johnson-Lindenstrauss was originally proven for Gaussian weights, but Achlioptas [Achlioptas 2001] showed that unit-weighted projections do not jeopardize accuracy in approximating distances. Furthermore, [Li et al. 2006] showed that unit random weights for most purposes can be made very sparse with the following probabilities:

$$u_j = \begin{cases} 1 \text{ with probability } & \frac{1}{2\sqrt{p}} \\ 0 \text{ with probability } & 1 - \frac{1}{\sqrt{p}} \\ -1 \text{ with probability } & \frac{1}{2\sqrt{p}} \end{cases}$$

In sum, we get lower prediction-error, robustness, scalability, and better approximation to the maximum-separation vector by using random unit weights in CHIRP. Furthermore, by constructing 2D projections from these random 1D projections and using (possibly) non-convex set covers on them, we substantially outperform LDA and other classifiers when the normality assumption is not plausible.