CHIRP: A new classifier based on Composite Hypercubes on Iterated Random Projections [*]

Leland Wilkinson[†]
Department of Computer Science
University of Illinois at Chicago
leland.wilkinson@systat.com

Anushka Anand
Department of Computer Science
University of Illinois at Chicago
aanand2@lac.uic.edu

Dang Nhon Tuan
Department of Computer Science
University of Illinois at Chicago
tdang@cs.uic.edu

ABSTRACT

We introduce a classifier based on the L^∞ norm. This classifier, called CHIRP, is an iterative sequence of three stages (projecting, binning, and covering) that are designed to deal with the curse of dimensionality, computational complexity, and nonlinear separability. CHIRP is not a hybrid or modification of existing classifiers; it employs a new covering algorithm. The accuracy of CHIRP on widely-used benchmark datasets exceeds the accuracy of competitors. Its computational complexity is sub-linear in number of instances and number of variables and subquadratic in number of classes.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining

Keywords

Supervised Classification, Random Projections

1. INTRODUCTION

Supervised classifiers face some serious obstacles. First, there is the *curse of dimensionality*: nearest neighbors in high-dimensional spaces diverge exponentially with dimension. A second problem is *computational complexity*: polynomial-time algorithms are infeasible in high-dimensional spaces. A third problem is *separability*: for some point sets in a vector space (e.g., a 2D disk surrounded by a ring) there exists no homeomorphism that will allow separation of regions using hyperplanes in the codomain.

There have been numerous approaches to overcoming these problems, such as hybrid classifiers [15, 39] or reduction of dimensionality through principal components, k-means clustering [13], or random projections [42].

[*]Supported by NSF/DHS grant DMS-FODAVA-0808860.

[†]Systat Software Inc. and Department of Computer Science, University of Illinois at Chicago

Our algorithm is an integrated solution that addresses all of these problems. CHIRP is a classifier that 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 sets of projections and regions into a decision list for scoring new data points.

CHIRP is a nonparametric, ensemble classifier that does not need to be customized for each data set, making it ideal for handling diverse data sets without prior knowledge of their structure. Its average prediction accuracy over a wide range of data sets exceeds that of competitive classifiers, especially ones that have sensitive parameters like kernel types, bandwidths, pruning criteria, etc.

CHIRP is based on a visual analytics framework [6] using what we call Composite Hypercube Description Regions (CHDRs) that can be used to define local and large-scale structures.

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, we employ the L^∞ or *sup* metric:

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

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. 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:

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

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 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 VisClassifier[6] algorithm assembled a set of 2D projections and used CHDRs to capture human visual interactions in order to classify a high-dimensional dataset. Following those promising results, we have subsequently removed humans from the loop and we 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.

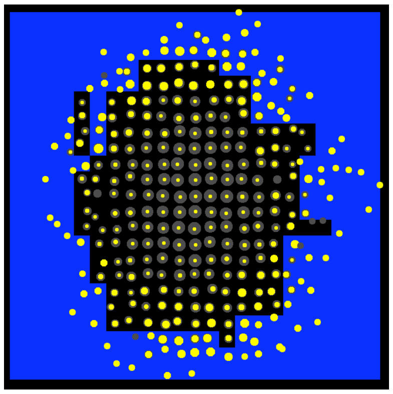

Figure 1: A CHDR covering class instances at the periphery of a projected random spherical distribution. Data are the Orange10 dataset from [19]. The data have been binned into a 24×24 grid with the size of each dot proportional to the count of instances in each bin. Yellow is used to represent the currently 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 [12]. 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 (with diminishing gains), but at the expense of time.

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 [10, 30]. 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 [25, 14, 21]. 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 [34], mixed integer programming [41], set covering [28], and decision lists [35, 4]. 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 [5, 3, 11, 16, 17, 29]. 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 Processing (Transforming and Normalizing)

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 next transform extremely skewed variables. We compute standardized skewness and kurtosis on each raw variable and sort the standardized values. We then sequentially test each element in these lists by adjusting the 99th percent critical value drawn from the standard normal distribution in order to control the false discovery rate [8]. If a standardized skewness or kurtosis value exceeds this adjusted critical value, we apply our transformation.

Our transformation is a folded square root:

$$t(x) = sgn(x)sqrt(abs(x))$$

This flexible transformation accommodates instances such as non-negative, positively-skewed data based on log-normal, Poisson, gamma, or exponential processes (counts, incomes, etc.), as well as long-tailed data from double-exponential, Cauchy, and similar distributions.

Our goal in transforming is to improve our chances of discovering class separation in relatively dense regions. This is especially important because we will use binning (a form of segmentation) to compress our data. Without transformation, highly-skewed 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.

After transforming, we normalize the data by rescaling each variable (feature) to the unit interval. 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 will be binning 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.

Before projecting, however, we need to scale categorical variables in order to project them into the same subspace with continuous variables. To scale categorical variables, we use a strategy derived from the latent class model [24]. 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 [20] and [27].

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 [19, 38]. 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 [37]. 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 [32, 44]. 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 status: 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 then have 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

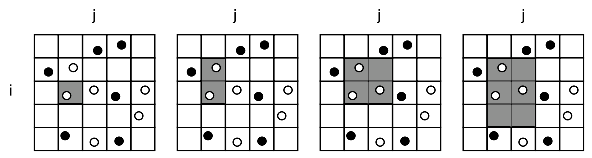

Figure 2: Growing a Hypercube Description Region (HDR) on a binned 2D projection. Each point is located at the centroid of the instances in each cell. Hollow symbols represent bins containing only instances of the current class. Solid symbols represent bins containing at least one instance of another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

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 grow a CHDR on a given 2D bin matrix with a recursive algorithm. Figure 2 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 ($b_{.,j+1}$), rightward ($b_{i+1,.}$), downward ($b_{.,j-1}$), and leftward ($b_{i-1,.}$) 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 begins to decrease. This strategy tends to result in squarish rectangles that cover pure or empty bins, similar to an approach in [3].

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 [40].

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. We terminate iterations when no CHDR can be constructed according to our rules.

4. CHIRP SCORING

To score a new instance, we transform and rescale 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 [33], this point-to-rectangle distance is asymptotically a nearest-neighbor statistic.

This scoring algorithm is based on a decision list [31]. 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 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 [7], [19], and other sources. Table 1 summarizes prominent aspects of these datasets.

5.2 Challenges

There are 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 (number of instances), large n, small p (number of variables), large p, ($n \ll p$) (Cancer), small g (number of groups), large g, relatively small training set (Poker, Segment) and disparate within-and-between-groups data densities (discrete, continuous, mixed, convex, non-convex (Orange10). 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. We should mention, in addition, that 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

Table 1: 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

the Weka data mining workbench [45]. 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 [23, 36, 1]. 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 [26], 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 3 summarizes the error performance for all classifiers. (Tables of these data are available on the senior author's website.) 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.

Figure 4 shows the performance of all the classifiers on these datasets. CHIRP is highlighted in red. No other family of classifiers (SVMs, Trees, ...) had the lowest or near-lowest error on as many datasets as did CHIRP.

5.4 Efficiency

CHIRP makes one pass through n rows of the training

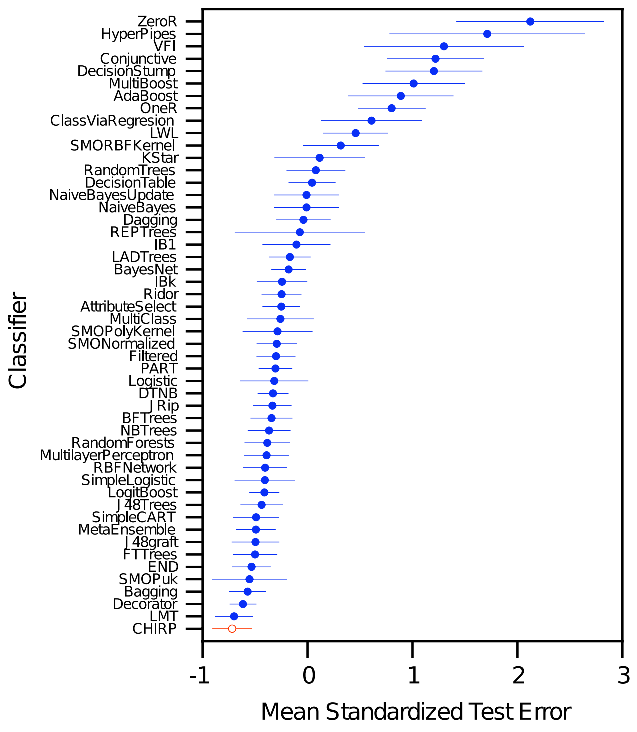

Figure 3: Average standardized error rates and associated 95% confidence intervals for CHIRP and Weka Version 3.6.1 classifiers. Default parameter values were used for all classifiers.

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 bin-matrix list and picks the top 5 candidate 2D bin matrices. It iterates through this process t times, adding a CHDR to the decision list at each step. Thus, we should expect

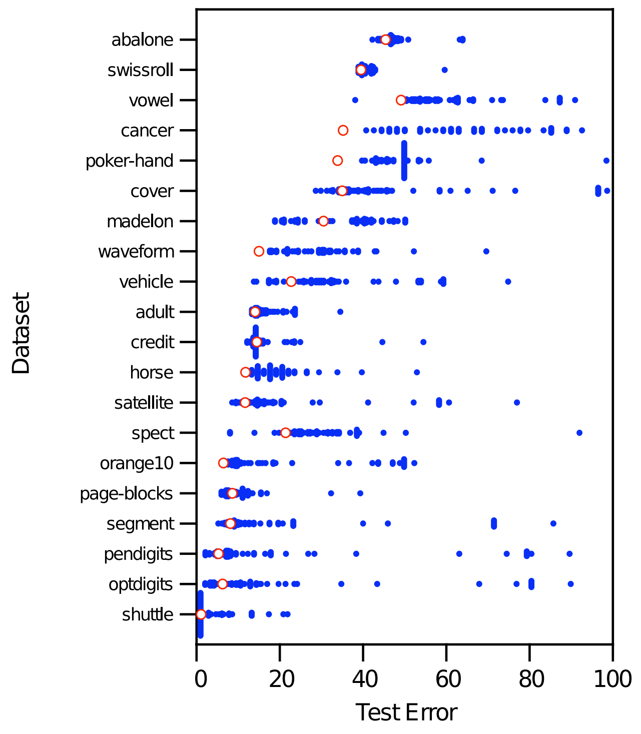

Figure 4: Errors for CHIRP (in red) and the other classifiers (in blue) on 20 test datasets.

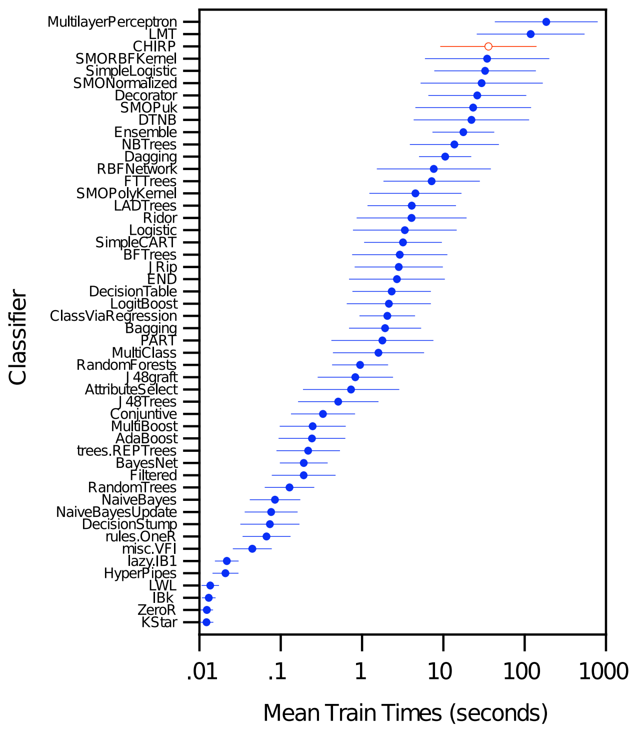

Figure 5: Mean training times and associated 95% confidence intervals for CHIRP and Weka Version 3.6.1 classifiers.

CHIRP to be $O(npgt)$ in time. To test this expectation, we did a simulation.

We generated Gaussians for $n = \{500, 5000, 50000\}$ and $p = \{20, 40, 60, 80, 100\}$, 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 $N(\mathbf{0}, \mathbf{I})$. We ran CHIRP once on each dataset.

The times are fit well ($R = .962$, with well-behaved residuals) with the simple linearized model:

$$E[\log(t)] = -10.34 + 0.94\log(n) + 1.66\log(g) + 0.68\log(p)$$

Our empirical results indicate that CHIRP is sublinear in n and p and subquadratic in g.

Figure 5 shows the training times for CHIRP and the other classifiers. Not surprisingly, some of the worst performing classifiers in Figure 3 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 in the middle of the pack. CHIRP's closest rival in performance, Logistic Model Trees, was significantly slower in training.

CHIRP is slower to train than many of the other classifiers in part because it is an ensemble classifier. In a parallel computing environment, each thread would run concurrently, so this would not be an issue. The longest training time (rules-DTNB on Madelon) was 38 hours. The longest training time for CHIRP was 2.6 hours on Poker Hand. The longest time for an SVM was 15 hours (SMOPuk on Adult).

Figure 6 shows the testing times per instance for 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 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 an instance was 4 minutes (lazyKStar on Satellite). The longest scoring time for CHIRP was 2 seconds (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.

6. DISCUSSION

Our experiment provides clear evidence that CHIRP outperforms competitive classifiers across a wide range of datasets. We intend to pursue the question of why this is true in further research, 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 theoretically how these components interact. We also need to investigate whether extending projections to 3D can improve performance without sacrificing scalability.

We do have several preliminary answers to the question of why CHIRP works so well, however. 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

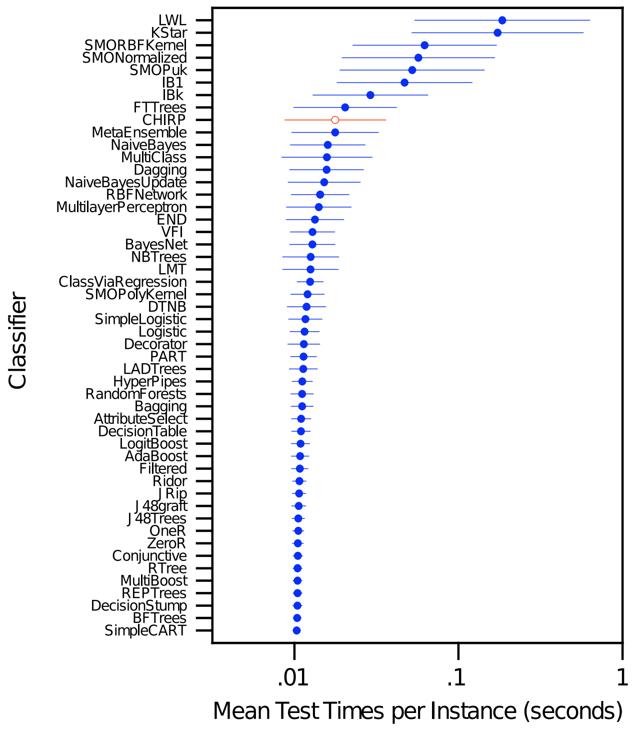

Figure 6: Mean testing times per instance and associated 95% confidence intervals for CHIRP and Weka Version 3.6.1 classifiers.

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, its 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. "Good" 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 as 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. Our future direction is to leverage the 2D aspect that motivated this approach to develop a new visualization to explore the "interesting" separating projections used by the classifier. With the strength and flexibility of CHIRP experimentally shown here, we can now use its framework to get a visual understanding of the class-separating relationships in the data and the inner workings of the classifier.

7. CONCLUSION

CHIRP is a relatively efficient classifier with average accuracy superior to other classifiers commonly associated with good performance on benchmark datasets. Its virtues are:

- Categorical variables expend only one degree-of-freedom. We scale categorical variables on each iteration, so there is no dummy-coding to inflate dimensionality.
- The performance is linear in complexity on n or p. It is subquadratic on g (the number of classes), but solution times are practical for up to a hundred 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 tiny. Its JAR file is under 50K in size. The algorithm iterates over three simple 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 *a priori* knowledge of the structure of the dataset.

8. REFERENCES

[1] M. R. Abdullah, K.-A. Toh, and D. Srinivasan. A framework for empirical classifiers comparison. In *Industrial Electronics and Applications*. IEEE, 2006.

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

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD*, pages 94–105, 1998.

[4] J. Aguilar, J. Riquelme, and M. Toro. Decision queue classifier for supervised learning using rotated hyperboxes. In *Proceedings of the 6th Ibero-American Conference on AI: Progress in Artificial Intelligence*, volume 4045 of *Lecture Notes in Computer Science*, pages 326–336. Springer, 1998.

[5] B. Alpern and L. Carter. The hyperbox. In *Proceedings of the IEEE Information Visualization 1991*, pages 133–134, 1991.

[6] A. Anand, L. Wilkinson, and D. N. Tuan. An L-infinity norm visual classifier. In *ICDM*, pages 687–692, 2009.

[7] A. Asuncion and D. Newman. UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 2007.

[8] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series* B, 57:289–300, 1995.

[9] P. Bickel and E. Levina. 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, 2004.

[10] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[11] S. Bu, L. V. S. Lakshmanan, and R. T. Ng. MDL summarization with holes. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 433–444. VLDB Endowment, 2005.

[12] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of AI Research*, 2:263–286, 1995.

[13] C. Ding and X. He. K-means clustering via principal component analysis. In *ICML '04*, page 29, New York, NY, USA, 2004. ACM.

[14] T. E. Flick, L. K. Jones, R. G. Priest, and C. Herman. Pattern classification using projection pursuit. *Pattern Recognition*, 23:1367–1376, 1990.

[15] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT '95*, pages 23–37, London, UK, 1995. Springer-Verlag.

[16] B. Gao. *Hyper-rectangle-based discriminative data generalization and applications in data*. PhD thesis, Simon Fraser University, 2002.

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

[18] Y. Guo, T. Hastie, and R. Tibshirani. Regularized discriminant analysis and its application in microarrays. *Biostatistics*, 1:1–18, 2005.

[19] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2001.

[20] C. Hegde and R. Baraniuk. Random projections for manifold learning. In *NIPS 2007: Proceedings of the 2007 conference on Advances in neural information processing systems*, Cambridge, MA, USA, 2007. MIT Press.

[21] L. O. Jimenez and D. A. Landgrebe. Projection pursuit for high dimensional feature reduction: paralleland sequential approaches. In *Geoscience and Remote Sensing Symposium, 1995. IGARSS '95*, volume 1, pages 148–150, 1995.

[22] W. B. Johnson and J. Lindenstrauss. Lipschitz mapping into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[23] R. King, C. Feng, and A. Sutherland. Statlog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, 9, 1995.

[24] P. F. Lazarsfeld and N. Henry. *Latent Structure Analysis*. Houghton Mifflin, Boston, 1968.

[25] E.-K. Lee, D. Cook, S. Klinke, and T. Lumley. Projection pursuit for exploratory supervised classification. *Journal of Computational and Graphical Statistics*, 14:831–846, 2005.

[26] P. Li. Robust LogitBoost and Adaptive Base Class (ABC) LogitBoost. In *UAI 2010 Proceedings*. IEEE, 2010.

[27] P. Li, T. J. Hastie, and K. W. Church. Very sparse random projections. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 287–296, New York, NY, USA, 2006. ACM.

[28] M. Marchand and J. Shawe-Taylor. The set covering machine. *Journal of Machine Learning Research*, 3:723–746, 2002.

[29] K. Q. Pu and A. O. Mendelzon. Concise descriptions of subsets of structured sets. *ACM Transactions on Database Systems*, 30(1):211–248, 2005.

[30] J. R. Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1993.

[31] R. L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

[32] D. W. Scott. On optimal and data-based histograms. *Biometrika*, 66:605–610, 1979.

[33] B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, New York, 1986.

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

[35] M. Sokolova, N. Japkowicz, M. Marchand, and J. Shawe-taylor. The decision list machine. In *Advances in Neural Information Processing Systems 15*, pages 921–928. MIT Press, 2003.

[36] A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, 21(5):631–643, 2005.

[37] H. A. Sturges. The choice of a class interval. *Journal of the American Statistical Association*, 21:65–66, 1926.

[38] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series* B, 58:267–288, 1995.

[39] R. Tibshirani and T. Hastie. Margin trees for high-dimensional classification. *Journal of Machine Learning Research*, 8:637–652, 2007.

[40] J. Tukey. A quick, compact, two-sample test to Duckworth's specifications. *Technometrics*, pages 31–48, 1959.

[41] F. Üney and M. Türkay. A mixed-integer programming approach to multi-class data classification problem. *European Journal of Operational Research*, 173:910–920, 2006.

[42] S. S. Vempala. *The Random Projection Method*. American Mathematical Society, Providence, RI, USA, 2004.

[43] H. Wainer. Estimating coefficients in linear models: It don't make no nevermind. *Psychological Bulletin*, 83(2):213–217, 1976.

[44] M. P. Wand. Data-based choice of histogram bin width. *The American Statistician*, 51(1):59–64, 1997.

[45] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham. 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*, pages 192–196, 1999.

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 $\boldsymbol{\Sigma}$. 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 $\mathbf{x}$ to the group with the smaller Mahalanobis distance

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

Equivalently, if the Fisher linear discriminant function

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

is negative, we assign $\mathbf{x}$ to the first group, otherwise to the second.

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

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

for our classification rule.

When $p > n$, the maximum likelihood estimate of $\boldsymbol{\Sigma}$ 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 $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$. 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 [9] 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 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*

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

For $p = 50$, for example, using $d_U(\mathbf{x})$ instead of $d_F(\mathbf{x})$ will shrink the values of $\hat{\boldsymbol{\mu}}_\mathbf{1}$ and $\hat{\boldsymbol{\mu}}_\mathbf{2}$ projected on $d_F(\mathbf{x})$ toward zero by a factor of approximately .3. The gains from this type of shrinkage are discussed in [43, 19, 38, 18] 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 [22]. 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 [2] showed that unit-weighted projections do not jeopardize accuracy in approximating distances. Furthermore, Hastie et al. [27] 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.