Decision Trees for Functional Variables

Suhrid Balakrishnan
Department of Computer Science
Rutgers University
Piscataway, NJ 08854, USA

David Madigan
Department of Statistics
Rutgers University
Piscataway, NJ 08854, USA

Abstract

Classification problems with functionally structured input variables arise naturally in many applications. In a clinical domain, for example, input variables could include a time series of blood pressure measurements. In a financial setting, different time series of stock returns might serve as predictors. In an archeological application, the 2-D profile of an artifact may serve as a key input variable. In such domains, accuracy of the classifier is not the only reasonable goal to strive for; classifiers that provide easily interpretable results are also of value. In this work, we present an intuitive scheme for extending decision trees to handle functional input variables. Our results show that such decision trees are both accurate and readily interpretable.

1 INTRODUCTION

We present an extension to standard decision trees (for example CART, Breiman et al. 1984 or C4.5, Quinlan 1993) that enables them to be applied to classification problems with functional data as inputs. In so doing, we aim to leverage the interpretability of decision trees as well as their other important benefits like reasonable classification performance and efficient associated learning procedures.

The application that motivated this work concerns vaccine efficacy. The study in question followed thirty vaccinated non-human primates (NHPs) for a year and then "challenged" the animals. Of the 30 NHPs, 21 survived the challenge and 9 died. Repeated measurements during the year assessed over a dozen aspects of the putative immune response. These measurements include an immunoglobulin G enzyme-linked immunosorbent assay (IgG), various interleukin measures (IL2, IL4, IL6), and a so-called "stimulation index" (SI), to name a few, with the number of measurements varying somewhat from animal to animal. The goal of the study is to understand the predictive value of the various assays with respect to survival.

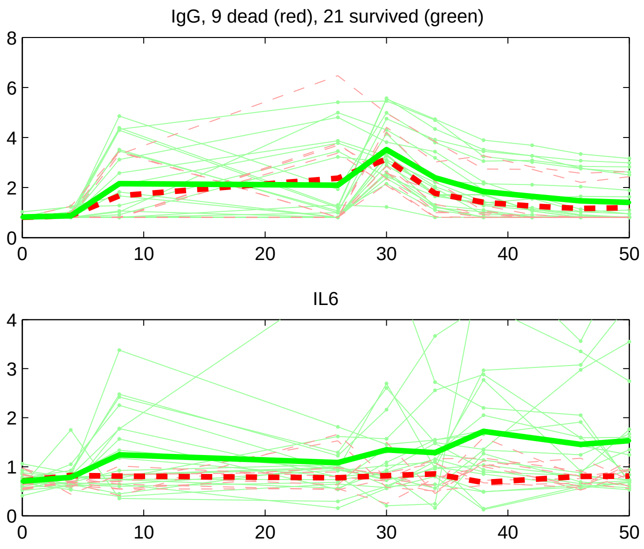

Figure 1: IgG and IL6 measurements for all 30 NHPs. Green (solid) represents animals that survived; red (dashed) represents animals that died. The thick curves represent sample means.

Our initial approach to this problem used a logistic regression model (Genkin et al., 2004) and treated each assay-timepoint combination as a separate input variable. While this provided good predictive performance, it selected a biologically meaningless set of predictors such as IgG at week 46, SI at week 38, and IL6 at week 12. The study immunologists instead sought insights such as "IgG trajectories that rise more rapidly after the 4-week booster shot and fall more slowly after the 26-week booster lead to higher survival probability." In other words, the underlying biology suggests that the shape of the assay curves should be predictive of survival rather than measurements at specific timepoints. Figure 1, for example, shows IgG and IL6 trajectories. For IgG, comparison of the thick green curve with the thick red curve shows that higher values are beneficial at the beginning, then lower values, and then higher values again towards the end. For IL6, it appears higher values of the curve in general are beneficial. We shall return to the motivating example in later sections.

More generally, we consider the following multi-class classification problem: the training dataset comprises n labeled training examples, $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i = [\mathbf{x}_{i1}, \ldots, \mathbf{x}_{id}]$ is a list of d features and $y_i \in \{1, \ldots, c\}$, the c labels. We account for functionally structured input data, by allowing the elements of $\mathbf{x}_i$ to be vectors/lists themselves, each representing an instance of a function of some independent variable. For example, a time series classification problem with only one time-series (or functional) variable of fixed-length T say, as input, would be represented as $\mathbf{x}_i = \mathbf{x}_{i1}$ in our setup, with the single feature $\mathbf{x}_{i1} = [x_{i1}^{(1)}, x_{i1}^{(2)}, \ldots, x_{i1}^{(T)}]$. Here time would be the independent variable.

We allow the inputs to be multivariate—meaning there may be more than one functional variable (i.e., more than one vector/list element of $\mathbf{x}_i$) and also allow for standard (non-functional) discrete/continuous/categorical variables (in this case the relevant components of $\mathbf{x}_i$ will be the corresponding scalars/nominal values). Thus, standard decision trees can be considered a type of special case of the above formulation where all inputs are restricted to length 1.

2 PREVIOUS STUDIES

Past approaches to this problem applying standard machine learning algorithms have typically relied on some sort of *ad hoc* and domain specific preprocessing to extract predictive features. A few previous studies look for interesting "events" in the training instances of the functional variables and then construct auxiliary variables based on them. These auxiliary variables are then used by either a particular classifier (decision trees, regression trees and 1-nearest neighbor, Geurts 2001 which retains interpretability), or generic classifiers (Kadous and Sammut, 2005) (with interpretable results available if a rule learning classifier is applied), or used in literals as base classifiers combined via boosting (Gonzalez and Diez, 2004). Another approach is the scheme by Kudo et al. (1999), which constrains the functional variables to pass through certain regions in (discretized) space and disallows other regions. There are also techniques that create features via specialized functional PCA techniques, designed to deal with large data applications (EEG data) where the functional variables are lengthy and numerous. Finally, there are support vector machine based approaches that are typically applied by defining an appropriate kernel function for the problem domain (Shimodaira et al., 2001).

3 CANDIDATE SPLITS FOR FUNCTIONAL VARIABLES

In order for a decision tree to be able to process functional variables, we first need to define candidate splits for such variables. In other words, we need to define a procedure that results in a partition of the space of possible function instances (in a manner similar to partitions for discrete/continuous/categorical variables). We describe the idea using the time series example. Consider a binary classification problem, i.e., $y_i \in \{1, 2\}$, and a functional input variable, a time series of fixed length T, $\mathbf{x}_{i1} = [x_{i1}^{(1)}, x_{i1}^{(2)}, \ldots, x_{i1}^{(T)}]$. While many splitting rules can be imagined, we propose the following: consider two representative curves, $\mathbf{x}_r$ and $\mathbf{x}_l$ where $\mathbf{x}_r = [x_r^{(1)}, x_r^{(2)}, \ldots, x_r^{(T)}]$ (and $\mathbf{x}_l$ is similarly defined). The (binary) split is defined by the set of function instances that are closer (in terms of some distance) to one representative curve than the other.

Note that this definition allows for the construction of very flexible partitions. Multi-way splits are an immediate extension and are quite simply defined by considering more than two representative curves. The distance function used can be application specific. In our experiments, we primarily focus on binary splits and two kinds of distance measures: Euclidean distance and dynamic time warping (DTW) distance. The use of DTW distance further exemplifies how flexible this kind of split is, enabling function instances of different lengths to be compared.

Classification with such splits in trees proceeds exactly as with standard decision trees, and the input test function instance follows the branch corresponding to the closest representative curve (with leaf nodes as usual holding predicted class labels).

Note that for a given distance measure, different choices for $\mathbf{x}_r$ and $\mathbf{x}_l$ lead to different candidate splits. Each candidate split corresponds to a partition of "function space" into two regions, functions within one region being more similar to each other than to functions in the other region. Our proposed approach rests on two basic assumptions. First, the partition should be interpretable. That is, the choice of $\mathbf{x}_r$ and $\mathbf{x}_l$ and the particular distance measure should lead to sets of functions that correspond to recognizably homogeneous sets of curves. Second, the true classification rule needs to be smooth with respect to the chosen distance measure. That is, functional inputs that are close together according the distance measure should generally belong to the same output class. What we attempt to show in later sections is that straightforward choices for $\mathbf{x}_r$, $\mathbf{x}_l$, and the distance measure lead to functional-split-enabled decision trees with good classification accuracy and tree structures that provide valuable insights.

3.1 FINDING REPRESENTATIVE CURVES

Having provided the intuition for defining the candidate test on the basis of proximity of the function instances to representative curves $\mathbf{x}_r$ and $\mathbf{x}_l$ (quantified in some manner), we now focus on how these curves can be automatically obtained from the data.

Note that the regularity assumption we make, is equivalent to assuming that the instances (or curves) cluster in some manner in the input domain. Consequently, a simple idea is to perform a functional variable-specific clustering, and then use cluster representatives in the candidate tests. The choice of different clustering procedures may lead to different decision trees and in general this choice will require some application-specific consideration.

In our experiments we used two clustering procedures: standard k-means clustering ($k = 2$ for binary partitions) with the Euclidean distance function between two instances of the same length, and a clustering procedure using DTW distances between instances. With DTW distance, the mean of the curves is not a particulary well-motivated representative of a cluster (instances look more like warped/time-shifted versions of each other than the mean). We instead use an instance as representative of the cluster—in particular, we perform the following EM-like iterations to find the representatives: a) Set the cluster representative to be the instance which is closest (has smallest combined DTW distance) to all the other instances in the cluster. b) Reassign instances to clusters based on their distance to the representatives[1]. We will refer to these procedures as "Euclidean clustering" and "DTW clustering" in the remainder of the paper.

3.2 CHOOSING GOOD SPLITS

While reasonable clustering procedures can provide reasonable representative curves, most standard clustering algorithms are only guaranteed to converge to locally optimal solutions. The standard approach to alleviate this problem is to do multiple restarts (multiplicity m) initialized randomly and pick the tightest clusters found.

Recall that in our application, however, the criterion for "goodness" of a candidate test is not how tight the found clusters are, but rather how well the representative curves partition the data class labels. Typical measures of partition purity used include entropy, information gain and the Gini diversity index. In our experiments we use the Gini index.

Summing up, in order to find good (high purity) splits for a functional variable we perform multiple restarts of clustering with random initializations (setting $m = 1000$, unless noted otherwise) and pick the partition (the representative curves summarize/define this partition) that has highest Gini index. This search procedure can be easily plugged in to standard divide and conquer decision tree building algorithms like C4.5 and CART, and Algorithm 1 provides an outline in high level pseudo-code.

Algorithm 1: Search procedure for functional variable split

```
Data: subset of the training data $D$.
Result: Representative curves $\mathbf{x}_r, \mathbf{x}_l$ that partition
        the input data $D_l, D_r$ ($D = D_l \cup D_r$), score
        of partition.
Initialize $best = [0, \phi, \phi]$ (stores $[score, D_l, D_r]$).
$\mathbf{x}_r, \mathbf{x}_l = \phi$.
for $j = 1, 2, \ldots, m$ do
    Run clustering procedure with random
    initialization. Obtain candidate representative
    curves and partition.
    Compute score (e.g., Gini index using candidate
    partition).
    if score is better than current best score then
        Update $best, \mathbf{x}_r, \mathbf{x}_l$.
    else
        Continue.
    end
end
```

4 APPLICATIONS

We now describe applications of our algorithm to both simulated and real datasets (see Table 1 for details). We will first examine three simulated datasets, the cylinder, bell, funnel dataset (CBF), an extension of it we created (CBF-2), and the control chart (CC) dataset. Table 2 provides some details about the predictive performance experiments. For each dataset,

[1]Note that this procedure bears resemblance to complete-link hierarchical clustering.

Table 1: Dataset Descriptions

Data	Src.	n	d_f	c	T_r	Protocol
CBF	*	798	1	3	128	10-fold CV
CBF-2	*	798	2	2	128	10-fold CV
CC	1	600	1	6	60	10-fold CV
JV	1	370	12	9	7-29	test 270
Bone	2	96	1	2	100	leave one out
NHP	*	30	7	2	7-10	leave one out

*: Own/Simulated, 1: UCI KDD Archive (Hettich and Bay, 1999), 2: Ramsay and Silverman (2002). n: Num. of observations, d_f: Num. of functional variables, c: Num. of classes and T_r: length of the functional instances.

we compare predictive error rates to one or more of the following: 1. **LR** (an L_1 constrained logistic regression classifier), using BBR[2] (Genkin et al., 2004) (for binary classification problems) and BMR[3] (for multi-class problems) trained with 10-fold CV used to pick the hyperparameter from amongst the set $\{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$. This is a reasonably fair baseline that represents state-of-the-art classifier performance (see Genkin et al. 2004, for how BBR compares to SVMs etc.). 2. **Seg.** (segmented inputs), which are the best previously published results from Geurts (2001) on various classifiers that use segmented auxiliary variables as input. 3. **Fnl.** (functional), which are the best previously published results of Geurts (2001) using comparably interpretable classifiers constructed by combining functional patterns and decision trees. 4. **Best**, which are the best published results otherwise known (not any of the other three categories).

For datasets where 10-fold CV was used to estimate error rates (CBF, CBF-2 and CC), the functional decision trees (**FDT**) were pruned by training on 8 out the 10 folds, and picking the sub-tree of the full tree that gave smallest error on the 9th fold (the pruning set). Finally, prediction errors were counted on the remaining 10th fold. For leave-one-out protocol datasets, no pruning was done.

Also, in order to display the functional splits we will use the following conventions in displayed trees throughout the paper: a functional split will be displayed by showing the functional variable name, a $<$ symbol, followed by a unique integer for the split. For example, x1 $<$ 1 represents a split on the functional variable 1 ($\mathbf{x}_{i1}$), and the index 1 likely indicates this is the root split. Further, for any split, **the left branch representative $\mathbf{x}_l$** will be shown in plots by a **solid line** and the **right branch representative $\mathbf{x}_r$**, will be shown by a **dashed line**.

[2]http://www.stat.rutgers.edu/~madigan/BBR
[3]http://www.stat.rutgers.edu/~madigan/BMR

4.1 CBF

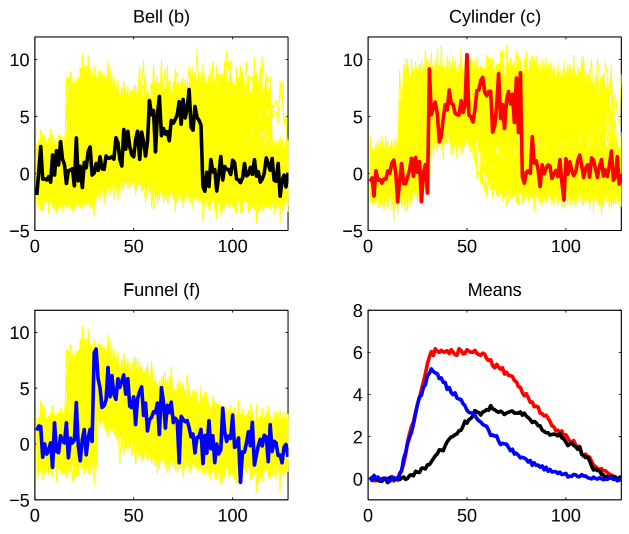

Figure 2: Cylinder, bell, funnel dataset.

The cylinder, bell, funnel dataset proposed by Saito (1994) is a three class problem $y_i \in$\{b,c,f\}, with one time series (functional) variable of fixed length. The equations describing the functional attribute for each class have both random noise as well as random start and end points for the generating events of each class, making for quite a lot of variability in the instances. As in past studies, we simulated 266 instances of each class to construct the dataset. Instances of each class are shown in Figure 2, where also shown is a particular instance of each class in bold and the computed class means in the bottom right panel. Although this

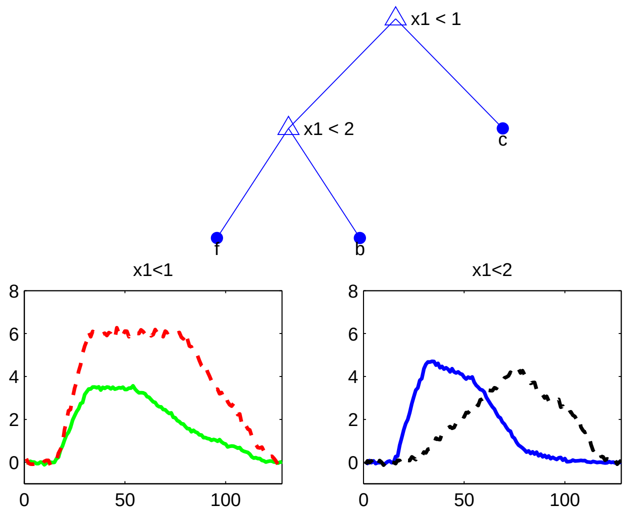

Figure 3: CBF results: pruned tree, splits.

is essentially an easy classification problem (reported accuracies are in the high 90's), it is often used as a sanity check when testing algorithms that perform functional variable classification. As far as predictive performance goes, our procedure also performs at par with some of the best known methods on this dataset (see Table 2). The functional decision tree provides a highly interpretable representation. Shown in Figure 3 is a pruned decision tree constructed on the whole dataset using Euclidean distance[4]. The leaf splits are, as expected, very representative of the class means (cf., Figure 2).

4.2 CBF-2

This is an artificial dataset we created that extends the CBF dataset by adding another independent functional variable. This second functional variable we also set to be a cylinder, bell or funnel instance. Finally, we create a binary classification problem with these inputs by assigning patterns to be class 1 if and only if the first variable instance $\mathbf{x}_{i1}$, is a cylinder and the second variable instance $\mathbf{x}_{i2}$, is a bell (again, we simulated 266 instances of each class for each variable). The classification problem is pretty much of the same hardness as the original CBF problem (see the predictive results table). We choose to work with this dataset because it is a perfect example to apply functional decision trees to: a combination of both input functional instances carry the entire predictive signal. The interpretability of the learned tree highlights how effective functional decision trees can be—see Figure 4. The splits, one on each variable, correspond exactly to the true data generating mechanism, and this mechanism is evident in the plots of the functional splits.

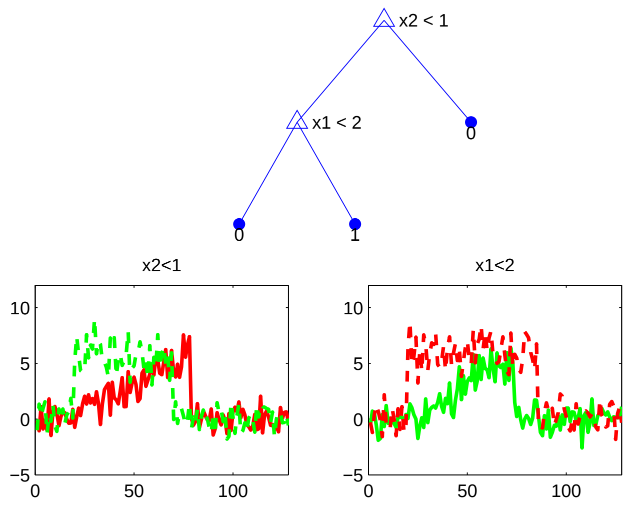

Figure 4: CBF results: learnt tree, splits. Branches predicting class 1 are shown in red.

[4]Note that the DTW results are reported in Table 2. Euclidean clustering results are slightly worse.

4.3 Control Chart

This is an artificial dataset in the UCI KDD Archive[5] consisting 100 objects of each of the six classes. The instances of each class $\in$\{normal,cyclic,up,down,increasing,decreasing\} are defined by 60 time points and the label broadly describes the behavior of the function with time—see Figure 5 (Up/Down denote a sudden jump in the series up/down respectively) . Euclidean clustering

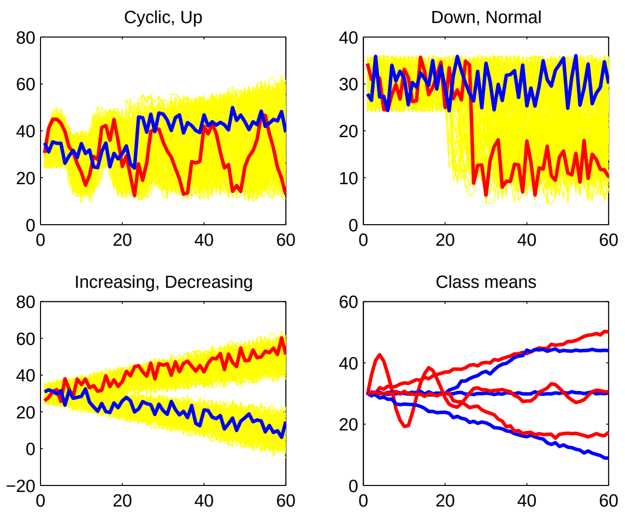

Figure 5: Control chart dataset.

performs best on this dataset, with competitive predictive accuracy to other techniques (Table 2). Figure 6 shows the complete functional decision tree along with select functional splits. Notice that the highest level split (x1 $<$ 1) corresponds broadly to partitioning the instances as those that generally increase and those that generally decrease (cyclic and normal are arbitrarily assigned to one or other). Appealingly, other internal splits display similar sorts of intuitive behavior (see x1 $<$ 4, for example) and leaf splits display strong representative instances of the component classes (see x1 $<$ 9 and x1 $<$ 10).

4.4 Japanese Vowels

Another dataset in the UCI KDD Archive, the Japanese Vowels dataset was first used in Kudo et al. (1999). The classification problem is a speaker identification task and the dataset consists the utterance of the Japanese vowels "a" and "e" by nine male speakers ($y_i \in \{1, \ldots, 9\}$). Each utterance is described by 12 temporal attributes, which are 12 time-varying LPC spectrum coefficients (for details of how these attributes were obtained see Kudo et al. 1999). An important challenge this dataset poses to many standard

[5]Hettich and Bay (1999), `http://kdd.ics.uci.edu`

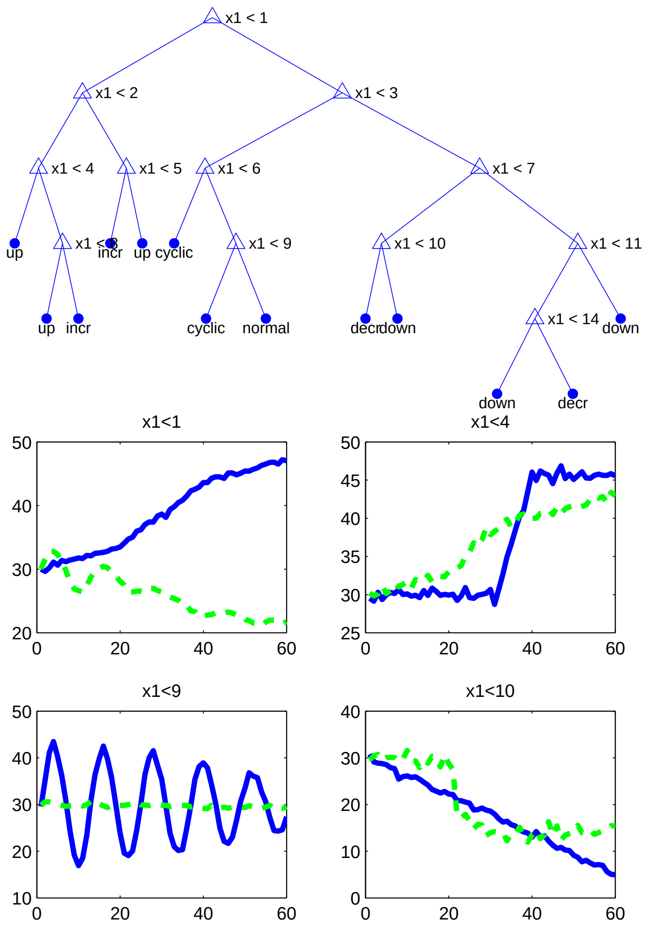

Figure 6: Control chart: learnt tree and some splits.

classification techniques is that the input sequences are of variable length (owing to the variable length of each utterance). For this reason, our results are obtained by applying the DTW clustering procedure, for which variable lengths are not a problem.

For this problem, since the number of classes is fairly large compared to the number of examples of each class, we constructed a 15-bit error correcting output code classifier (Dietterich and Bakiri, 1995) from the functional decision trees. Although our prediction accuracy isn't quite at par with the best methods for this problem (see Table 2) it is better than that of the other functional classifier that produces interpretable outputs, Geurts (2001).

More interesting for this dataset, are the learnt decision trees and their comparison with Kudo et al.'s analysis of the problem. Shown in Figure 7 is the tree learnt for speaker 1 (a 1-vs.-all binary classification problem), select splits from the tree, and previously published results based on Kudo et al.'s analysis (who also construct individual speaker classifiers). Kudo et al.'s procedure constructs regions in space through which the speakers curves mostly should (solid black rectangles) and mostly should not (dashed purple rectangles) pass. The qualitative correspondence between these regions and the functional splits we obtain is quite striking.

4.5 Bone

This second real dataset records the planar cross-sectional outline of the intercondylar notch from the knee joint (on the femur) of adults. The data has been collected from exhumed skeletons and includes concomitant information such as the sex of the individual. We obtained the data from Ramsay and Silverman's book (Ramsay and Silverman, 2002) data section[6]. The raw data (outline of the coordinates of the bone) is first parameterized by arc length and sampled—see Ramsay and Silverman (2002) for details of this standard pre-processing. This arc length parameterized data is what we set (both x and y coordinates) as our single two-dimensional functional variable (the independent variable being arc length). The binary classification problem involves predicting arthritis and past analyses of this problem have shown the notch shape to be an important predictor. Besides obtaining competitive predictive accuracy on this dataset, the learnt tree displays interesting splits. Firstly, the tree contains both non-functional splits (on sex of the individual) and functional splits. Second, the functional splits themselves are instructive about bone shape and arthritis implications—see Figure 8 (shown with a comparable plot from the literature). Corroborating the conclusions James and other authors reached by independent analyzes on the same dataset (James and Silverman, 2005; Ramsay and Silverman, 2002), the functional splits we learn show that both variability in the y-direction (shrunken bones) and bending to the right are predictive of a greater risk of arthritis.

4.6 NHP study

As described in the introduction, this dataset involves time series measurements (of different lengths) related to the state of the immune system of vaccinated NHPs. Once again, the tree we learn is competitive in predictive accuracy. Figure 9 shows the learnt tree and functional splits (as in the introduction, red corresponds to branches for predicting death $y_i = \mathrm{d}$, while green corresponds to NHPs that survive, $y_i = \mathrm{a}$). The splits are biologically sensible. A increasing IL6 trajectory is predictive of survival; A rapid early rise in SI is predictive of death; an early rise in TNF (tumor necrosis factor) is predictive of survival.

[6]`http://www.stats.ox.ac.uk/~silverma/fdacasebook/`

Table 2: Predictive Performance—Error Rates

Data	**Best**	**LR**	**Seg.**	**Fnl.**	**FDT**
CBF	0^1	2.77	0.5	1.17	0.13^D
CBF2	-	5.29	-	-	0.25^D
CC	0.33^2	10.33	0.5	2.33	2.0^E
JV	3.8^3	-	3.51	19.4	9.46^D
Bone	-	21.86	-	-	19.79^E
NHP	-	33.33	-	-	26.67^E

1: Kadous and Sammut (2005) 2: Geurts and Wehenkel (2005) 3: Kudo et al. (1999) D: DTW clustering, E: Euclidean clustering.

5 DISCUSSION, FUTURE WORK

In this paper, we presented a simple and effective extension to decision trees that allows them to operate on functional input variables. We presented results showing that these functional decision trees are accurate and produce interpretable classifiers.

Many extensions to the basic idea presented here suggest themselves; we describe a few. The representative curves can be generated by more sophisticated clustering algorithms; of particular interest would be ones designed for clustering functional curves. For example, the one proposed by James and Sugar (2003). Also, a range of algorithms from model-based clustering (e.g. HMM based) to non-parametric clustering (e.g. Gaussian processes based clustering methods) might be applied.

Further, one is not limited to decision trees as the base classifier either. An alternative way to view a single functional split is that it defines an auxiliary variable that may be used in addition to standard features in any classification algorithm. Multi-way splits, for example, might be particularly powerful features in multi-class problems. Finally, predictive accuracy can likely be improved by boosting these functional decision trees, a topic we are currently investigating.

Acknowledgements

We thank the US National Science Foundation for financial support as well as the KD-D group, who supported this work through NSF grant EIA-0087022.

References

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth International Group., 1984.

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

A. Genkin, D. D. Lewis, and D. Madigan. Large-scale bayesian logisitic regression for text categorization., 2004. URL http://www.stat.rutgers.edu/~madigan/PAPERS/.

P. Geurts. Pattern extraction for time-series classification. In L. de Raedt and A. Siebes, editors, *PKDD*, LNAI 2168, pages 115 – 127, Freiburg, September 2001.

P. Geurts and L. Wehenkel. Segment and combine approach for non-parametric time-series classification. In *PKDD*, October 2005.

C. J. A. Gonzalez and J. J. R. Diez. Boosting interval-based literals: Variable length and early classification. In A. Kandel M. Last and H. Bunke, editors, *Data mining in time series databases.* World Scientific, 2004.

S. Hettich and S. D. Bay. The UCI KDD archive, 1999. URL http://kdd.ics.uci.edu.

G. James and B. Silverman. Functional adaptive model estimation. *Journal of the American Statistical Association*, 100:565 – 576, 2005.

G. James and C. Sugar. Clustering for sparsely sampled functional data. *Journal of the American Statistical Association*, 98:397 – 408, 2003.

M. W. Kadous and C. Sammut. Classification of multivariate time series and structured data using constructive induction. *Machine Learning Journal*, 58: 179 – 216, 2005.

M. Kudo, J. Toyama, and M. Shimbo. Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20(11-13):1103 – 1111, 1999.

J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, California, 1993.

J. O. Ramsay and B. W. Silverman. *Applied Functional Data Analysis: Methods and Case Studies.* Springer-Verlag, New York, 2002.

N. Saito. *Local feature extraction and its application using a library of bases.* PhD thesis, Yale University, 1994.

H. Shimodaira, K. i. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *NIPS*, volume 2, pages 921 – 928, 2001.

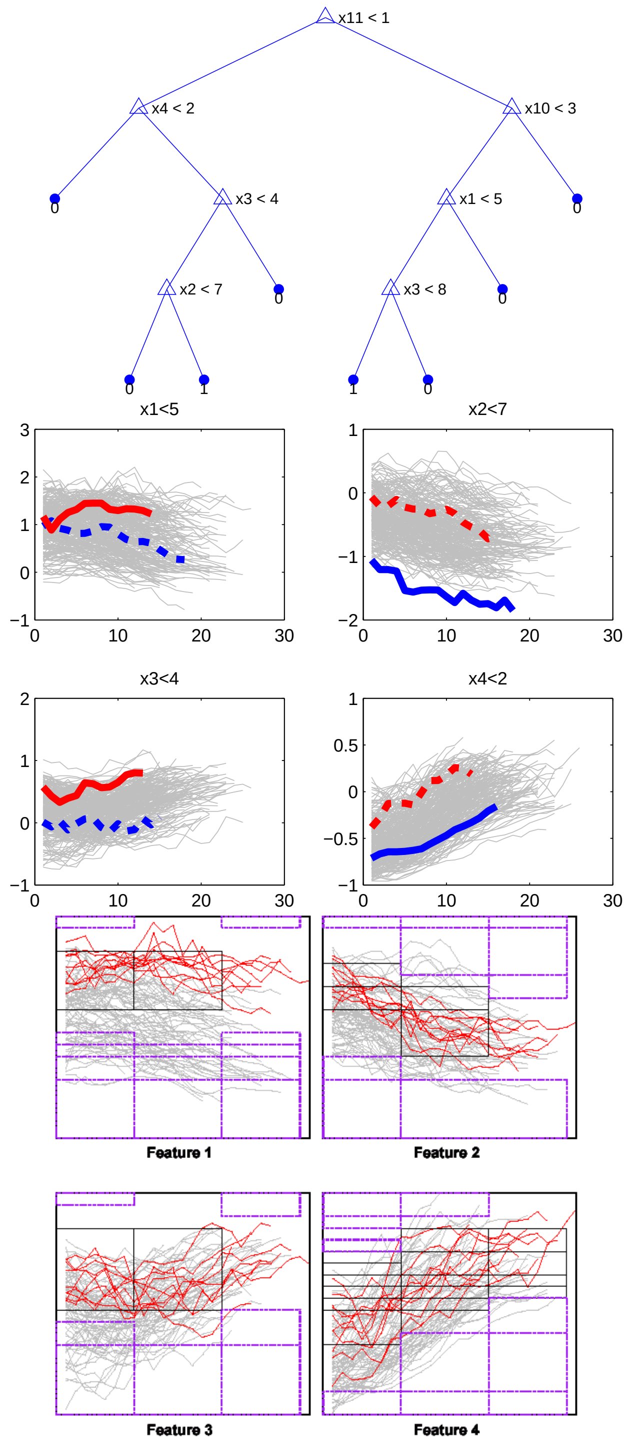

Figure 7: Japanese Vowels: Functional splits corresponding to reported results in Kudo et al. (1999). Branches corresponding to speaker 1 have been colored red for ease of comparison. Note: the bottom figure is a capture of a figure in Kudo et al. (1999).

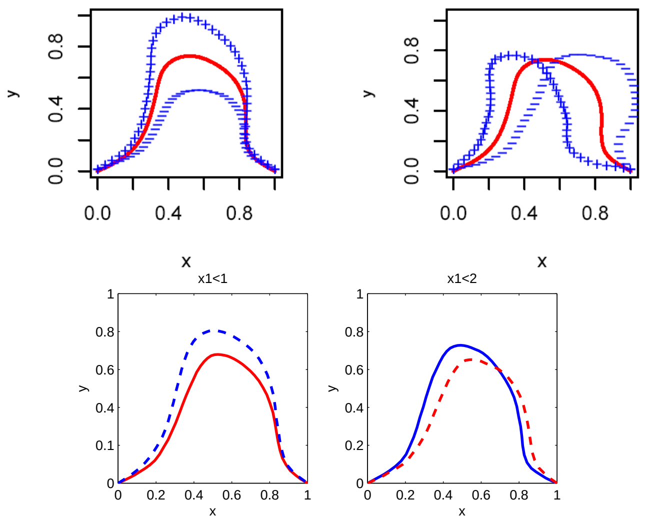

Figure 8: Bone data comparative results. Top Row figures: captures from a figure in James and Silverman (2005). Plot shows, in blue, the first two principal components of the predictive model they propose. '-' sign curve being for arthritic bones and the '+' curve being for healthy bone shapes (mean bone shape in red). Lower row figures: root and next level split of learnt FDT (tree not shown). Branches of the functional splits predicting arthritis are colored red.

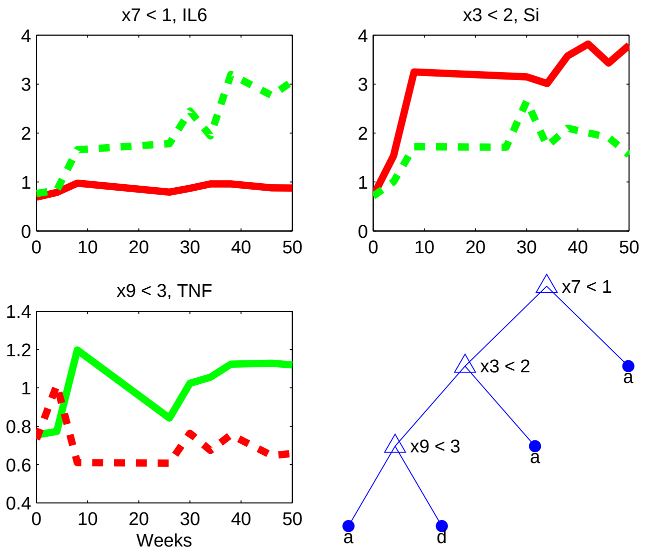

Figure 9: NHP learnt tree, functional splits. Branches of the functional splits predicting death are colored red.