Learning through spectral projectors

C. Avart[1], S. Bonnot[2], and T. Whalen[3]

[1]Analog Devices, Christian.Avart@analog.com
[3]Analog Devices, Thor.Whalen@analog.com
[2]University of São Paulo, Institute of Mathematics and Statistics, sylvain@ime.usp.br

***Abstract*—The tasks of sound recognition and classification have recently taken a front row through their applications to music (music recognition, beat detection) and voice (biometry, speech analysis). But other, industrial applications appeared as well. Machine health monitoring is certainly one of the most promising ones. For rotating or vibrating machines, an early detection of the various tears and faults makes all the difference. There is a need for fast and lightweight detection systems. This article describes one method that allows to treat a wide range of signals, while keeping a fast-learning and lightweight implementation, amenable to "edge" and even analog computations. We present here the main algorithmic component, dubbed *spectral projector*. We apply it to two very different contexts: rotating machines and Human Activity Recognition.**

***Index Terms*—Filter banks; Sound Analysis; Human Activity Recognition; Machine Health Monitoring; Principal component analysis; Linear Discriminant Analysis**

I. Introduction

The pipeline described in this article addresses supervised sound classification. The models presented here learn initially from a dataset of user-tagged waveforms and then classify new waves into one of the categories already seen. One use-case is predictive maintenance. An airline may want to predict when a non-critical component of an airplane needs to be replaced. Replacing it too late may incur costly repair and downtime, as would an unnecessarily early replacement. Through a microphone placed near the mechanical part in question, our algorithm listens to changes in the sound emission over time and communicates to the maintenance crew the status of the part, allowing for optimal replacement time. The model does not limit itself to sound analysis: any kind of vibration with energy in a frequency range detectable by a sensor can be used. The *spectral projector* we will describe is initially fed a (discretized) waveform, optionally annotated by the user with tags describing the relevant content of the sound.

We concentrate here on the data produced by accelerometers: located on some machine to be diagnosed, or located on some user for whom one wants to recognize the activity.

The whole pipeline is described by Figure 1.

A. Chunking of the signal

As a preliminary step, the waveform is cut by a *chunker* into shorter consecutive segments called chunks. At this step, a light pre-processing can be applied (like for example background noise removal for sounds applications). If necessary, one can also trim beginning and trailing silences. The chunking can be done blindly (i.e cut into consecutive non-overlapping chunks having a common duration) or adaptively (chunk when something interesting happens, like a sudden shock). The amount of overlapping can also be controlled (having zero-overlap as a default option).

The output of the chunker is then fed to the Featurizer.

B. Feature vectors

From each chunk produced by the chunker, we produced *spectral features*: the default option is to compute a FFT and take the norm of those vectors, but more involved treatments of the signals (using filter-banks) can be chosen at this point. The resulting spectra are considered as our initial (raw) feature vectors. Those raw feature vectors are typically high-dimensional. We usually want to project them to a much smaller subspace of the feature space. The various possible projections used are described later in more details.

C. Summarizing the data

This part is not used for the classification task, but we mention it to illustrate further aplications. Once the feature vectors have been projected, we can cluster the spectral projections. The resulting centroids can be seen as a useful concentrated summary of the dataset, that can be further used for anomaly detection.

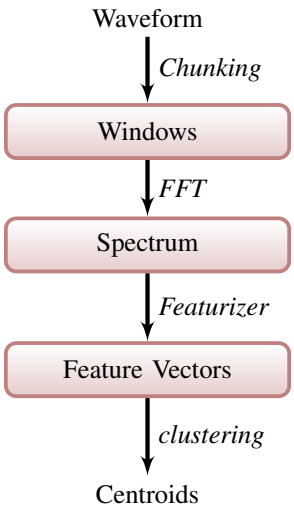

Fig. 1. Learning phase

D. Classification phase

Once the chosen *featurizer* has been applied to the raw feature vectors, one can start the classification phase. We compare various schemes and their respective accuracies.

II. Extracting meaningful features from waveforms

A. Featurizer

The various projections used can be divided into two broad classes: the simplest ones (modelled on filter banks) which perform a partition of the spectra into consecutive bins (of varying sizes), followed by an averaging of the bins, and the more refined ones allowing all types of (linear) projections of those spectra.

1) Desirable qualities of a featurizer: In broader terms, the quality of a *featurizer* can be investigated through its characteristics. Here are some desirable qualities that a featurizer should possess:

- **Independence:** to avoid redundancy, the created features should be independent from each other. For linear featurizers, one can simply aim for *linear independence* of the features. However, the independence of the features can come at the expense of the interpretability of those.
- **Extensibility**: an ideal featurizer should be parametrizable in such a way that, by tuning its parameters one could make it to act as a *lossless* featurizer if desired. An example would be the Principal Components Analysis: by choosing the number of components to coincide with the initial dimension of the feature space, the transformation obtained is invertible and no information is lost.
- **Fast to learn**;
- **Fast to compute**;
- **Reversible:** ideally, once a featurizer has been applied to a given collection of chunks of spectra, one should be able to revert the whole process and obtain an approximation of the initial stream signal. In the case where the initial signal is a sound waveform, one should then be able to listen to the transformed signal.

2) Band models: There is one class of linear projections, that we call *band models* in which the vector basis of the initial feature space is partitioned into various consecutive bins. Those bins can be of equal size (labelled **"Equal bins"** in the subsequent result tables) or of varying size. In this study, we concentrate on bins with sizes varying logarithmically ("**Log bins**"). These very simple band models offer many of the desired qualities of a featurizer:

- they create linearly independent features (being averages of two by two disjoint groups of vectors of the initial basis);
- they are fast to compute;
- they are *extensible* (in the above sense)

As an added bonus, they are easily amenable to analog computations.

3) Other projections: The various other projections considered are as follows:

1) **Random projections**: more precisely we used scikit-learn's "Gaussian Random Projection";
2) **PCA**: the standard principal component analysis projection,maximizing the variance;
3) **LDA**: the modified Linear Discriminant Analysis, maximizing the separation between classes (see below its description);
4) **NCA**: the implementation of Neighborhood Components Analysis made available by scikit-learn;
5) **unsupervised LDA**: a version of LDA where the initial classes are replaced by new ones obtained by clustering the dataset (using KMeans algorithm);
6) **unsupervised NCA**: a version of NCA where the initial classes are replaced by new ones obtained by clustering the dataset (using KMeans algorithm).

Chained projections.

Linear projections can be composed, and longer *chains* of such projectors can be made: one could imagine a situation where a PCA is first applied to the dataset, followed by an LDA. Chaining the projections can even solve a limitation of the standard LDA projection: by design, the standard LDA projection can only project

to a space with dimension less than $k-1$ (where k is the number of initial classes). We can overcome this limitation by using a *modified* LDA, where "chains" of such projections can be produced, each applied to a "residual" dataset as explained in the algorithm.

Algorithm 1 Chained projections

Require: Chain $(P_1, \ldots, P_m)$: chain of linear projectors

Initialize $k = 1$,
repeat
  compute the parameters of P_k
  apply P_k to the dataset
  compute the residue $r(X)$ of the dataset X (i.e $r(X) := X - P_k(X)$)
  reduce the dataset: $X \leftarrow r(X)$
  $k \leftarrow k + 1$
until num. features obtained or $k = m$

B. Classification

The various classification schemes we used are standard:

1) **Nearest Neighbors**: classification is made by voting, using the k nearest neighbors ($k = 10$),
2) **Linear SVM**: Linear Support Vector Classification,
3) **RBF SVM**: Radial Basis Function kernel for the support vector machine,
4) **Decision Tree**
5) **Random Forest**
6) **Neural Net**: Multi-layer Perceptron classifier,
7) **AdaBoost**
8) **Naive Bayes**
9) **LDA**

When we used a non-standard chained projection for LDA, the classification scheme employed is still the one used traditionally. Namely, we recall, following [2], that for each class k we can form the corresponding *linear discriminant function*

$$\delta_k(x) = x^T \mathbf{\Sigma}^{-1} \mu_k - \frac{1}{2} \mu_k^T \mathbf{\Sigma}^{-1} \mu_k + \log \pi_k$$

From those we define the *decision function* as

$$G(x) = \operatorname{argmax}_k \delta_k(x)$$

All the various parameters are deduced empirically from the dataset:

- $\hat{\pi}_k = N_k / N$, with N_k being the number of class- k observed
- $\hat{\mu}_k = \sum_{g_i = k} x_i / N_k$ the mean of the $k-$th group
- $\hat{\mathbf{\Sigma}} = \sum_{k=1}^{K} \sum_{g_i = k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T / (N - K)$

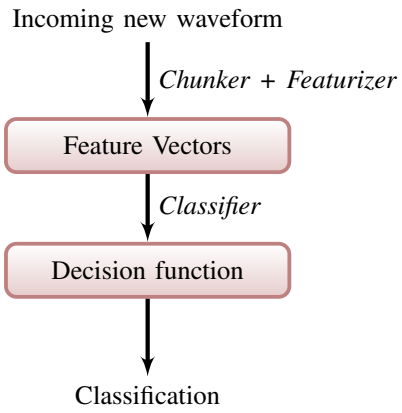

Fig. 2. Classification phase

III. Unconstrained vs constrained learning

In this section we briefly review some modifications that can be made to the models to make them lighter in terms of computational resources.

A. Possible constraints: the cost of learning

The global setting is as before: a continuous stream of data is observed for some time interval (for example, a waveform, accelerometer data, ...). In order to produce raw feature vectors, a spectrum of size m is computed on time windows of constant size T. The main goal is to find an optimal linear projection $p : \mathbb{R}^m \to \mathbb{R}^n$ towards a target space of fixed smaller dimension (say, $n = 15$ when $m = 1024$ for example), and possibly satisfying some constraints, imposed by practical considerations. For example, the range of data could be constrained by the devices used to perform the classification, or there might be memory limitations, forcing the projection matrix to be sparse, etc ...In particular, in the analog case, where computations are made at the level of the device, such constraints do appear.

Typically the space where we will perform optimization will be smaller than $\mathbb{R}^{n.m}$. The following is a simple list of possible constraints on the matrix M. Depending on the goal we want to optimize, only a subset of these will be required.

a) **Compact domain**: all entries of the matrix of size $n \times m$ are to be taken in some compact interval $[a, b] \subset \mathbb{R}$,
b) **Sparsity**: only a given fraction $f \in [0, 1]$ of the entries is allowed to be non-zero,
c) **Full Rank**: the matrix M must have rank n,
d) **Contiguity:** rows of the matrix M must present only one block of consecutive non-zero entries,

e) **Reduced overlap**: columns of M can have at most 2 non-zero entries,
f) **Band exclusion**: a given set S of "forbidden" columns can only be filled with zeroes.
g) **Consecutive bands**: the vectors in the standard basis of the domain are partitioned into m groups, and the projection matrix returns the averages of the coordinates corresponding to each group
h) **Binary matrix**: entries are in $\{0, 1\}$.

Expressing the constraints.

Let us quickly show how to reframe the determination of a projection matrix (say of PCA type) in the presence of such constraints. Let $\mathbf{X}$ be a training dataset of dimension $N \times d$, centered, made by the concatenation of sample vectors $\mathbf{x_1}, \ldots, \mathbf{x_N}$. From it we can compute the $d \times d$ empirical covariance matrix $\mathbf{A} = 1/N \cdot \mathbf{X}^\top \mathbf{X}$.

Recall that the *principal components* can be computed iteratively (i.e starting with the leading principal component vector) or as a group, by computing a matrix U satisfying the following:

$$\underset{U \in \mathbb{R}^{d \times k} : U^\top U = I}{\operatorname{argmax}} \operatorname{trace}\left(U^\top \sum_{i=1}^{N} \mathbf{x_i} \mathbf{x_i}^\top U \right)$$

Let us say that we want to consider a PCA objective with disjoint supports: all rows of the desired projection matrix must be non-negative, with sum equal to 1, and all non-zero entries in a row are equal. In addition, pairs of rows must have disjoint supports (meaning that no index has non-zero entries in both rows of the pair), which from the lemma is equivalent to requiring the orthogonality of the matrix.

For example one might consider:

$$M = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Let us denote by $\mathcal{S}$ the class of *simple matrices* such that for each row $i \in \{1, \ldots k\}$, the entries of it are either zero or equal to a positive value v_i.

When we restrict ourselves to non-negative entries (for example in the case where all entries are 1 or 0), there is a nice simple observation one can make:

Lemma 1. *For a matrix with non-negative entries, any two rows have disjoint supports if and only if they are orthogonal.*

Two vectors $\mathbf{u} = \sum \lambda_i \mathbf{e}_i, \mathbf{v} = \sum \mu_i \mathbf{e}_i$ expressed in the same basis $(\mathbf{e}_i)_{1 \leq i \leq n}$ have *disjoint supports* if $\lambda_i \mu_i = 0$ for each $1 \leq i \leq n$.

Then the optimization problem we want to consider amounts to finding a matrix U satisfying:

$$\underset{U \in \mathbb{R}^{d \times k} : U^\top U = I, U \in \mathcal{S}}{\operatorname{argmax}} \operatorname{trace}\left(U^\top \sum_{i=1}^{N} \mathbf{x_i} \mathbf{x_i}^\top U \right)$$

Observe that the orthogonality condition $U^\top U = I$ already implies the condition on disjoint supports, and also the condition that rows must sum to 1.

Approximating a PCA objective with disjoint consecutive supports Without fully resolving the optimization problem mentioned above, one can try to approximate a PCA projection with a *band matrix*. To give an example, let us say we want to partition the 1024 dimensions of the spectrum into 15 consecutive groups and average the values in each group. We want to determine the consecutive groups of basis vectors in a way that mimicks the PCA approach.

One possibility is to determine the projection matrix iteratively, i.e *band after band*. In other words:

1) Compute $\mathbf{v}_1$ the first main principal component vector (for the true unconstrained PCA matrix)
2) Find the top band $\mathbf{w}_1$ forming the smallest angle with v_1
3) (**Deflation step**) project the data on the orthogonal complement of $\mathbf{w}_1$ and go back to first step.

This greedy approach gives a fast approximation of the PCA projection by matrices of the desired format.

IV. Experimental results

A. Datasets used

1) Case Western dataset: This dataset has been widely used in the literature (see [6] for references). In this dataset, certain faults (with diameter from 0.007 to 0.028 in.) were made on the drive- and fan-end bearings of a motor. The motor was then run at constant speed (with value depending on the experiment, in the interval 1720 rpm to 1797 rpm). Acceleration was measured in the vertical direction of the drive-end bearing (DE), the fan-end (FE) and the base plate of the motor (BA). The sample rates were 12kHz and 48kHz. See [5] for more details and for the dataset itself.

For our own study, we used a subset of the entire dataset: the DE (drive-end) subset sampled at 48kHz and the Normal subset also sampled at 48kHz.

The failure types are of the type "prefix-diameter" where prefixes are: "inner ring (IR)", "Outer ring (OR)" and diameters are taken in the list $(07, 014, 021)$. The no-failure case is designated as "normal".

Dataset composition of CWRU.

RPM	Load	Fault Diam. (mms)	Setting
1772	1	7,14,21	A
1750	2	7,14,21	B
1730	3	7,14,21	C

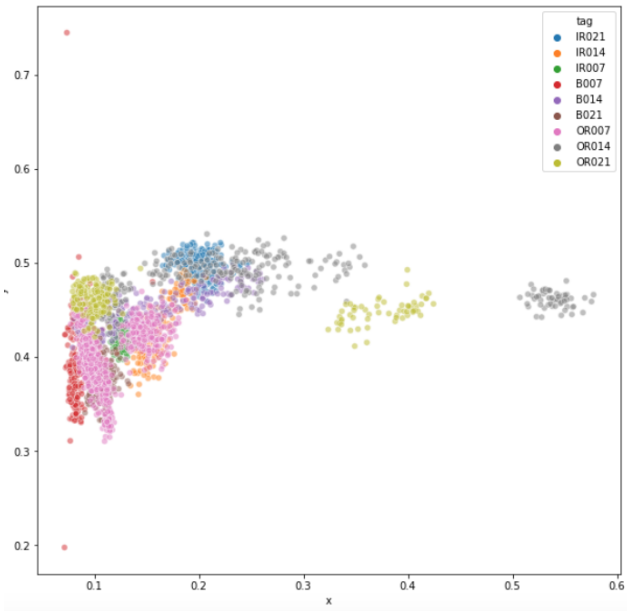

Fig. 3. 2D spectral projection

2) HAR dataset: The dataset we used was from [3], available at [4]. For general results on Human Activity Recognition see [7].

A group of 30 participants was equipped with smartphones (at waist-level). The accelerometer data were recorded while they were performing six basic activities: Walking, Walking Upstairs, Walking downstairs, Sitting, Standing, Laying. More precisely $3-$axial linear acceleration and $3-$axial angular velocity at a constant rate of 50Hz were recorded. The participants were then split into a training set (70% of the volunteers) and a testing set.

In the initial study, the dataset was preprocessed as follows: noise has been filtered, gravitational components of the acceleration has been separated (using a Butterworth low-pass filter with 0.3 Hz cutoff frequency). The signal was split into short fixed-width sliding windows of 2.56 sec and 50% overlap (with 128 readings/window).

From these windows a 561-feature vector with time and frequency domain variables was obtained.

For our own results, we used only the raw data and one single channel (acceleration along the X-axis). We tried separating as well the gravitational component, without improving the results.

B. Results

TABLE I
CASE WESTERN DATASET

	accuracy	fitting_time	classify_time
lda + Linear SVM	1.000000	3.797352	0.037898
lda + Nearest Neighbors	1.000000	3.476086	0.346804
lda + Neural Net	1.000000	3.498505	3.077243
nca + Nearest Neighbors	1.000000	42.344748	0.494811
lda + LDA	0.999638	3.566572	0.024349
nca + Neural Net	0.999638	35.136272	10.746401
lda + Naive Bayes	0.998552	3.461116	0.017265
Equal_bins + Nearest Neighbors	0.997828	0.059211	0.366956
pca + Nearest Neighbors	0.996381	0.726136	0.354215
nca + Linear SVM	0.996019	38.030710	0.297274
Equal_bins + RBF SVM	0.994209	0.047753	1.199371
Log_bins + RBF SVM	0.993123	0.045755	1.291479
Log_bins + Nearest Neighbors	0.993123	0.062100	0.327928
pca + Neural Net	0.991314	0.635781	6.822366
lda + Random Forest	0.991314	3.625565	0.106975
pca + Linear SVM	0.975389	0.661666	0.421153
Random_proj + Nearest Neighbors	0.954397	0.052976	0.881663
Random_proj + Neural Net	0.940644	0.040065	5.360062
Log_bins + Neural Net	0.937749	0.049572	5.535397
Equal_bins + Neural Net	0.937025	0.048455	6.701609
Log_bins + Random Forest	0.896127	0.050428	0.120809
Equal_bins + Random Forest	0.888527	0.049379	0.105116
nca + Random Forest	0.828085	38.084676	0.108608
pca + Random Forest	0.808180	0.654959	0.097239
lda + Decision Tree	0.800217	3.536657	0.035294
Equal_bins + Linear SVM	0.796960	0.053738	1.231600
pca + Decision Tree	0.781035	0.665487	0.083154
Log_bins + Linear SVM	0.778502	0.050175	1.316424
unsupervised_nca + Nearest Neighbors	0.771987	30.185886	0.567682
unsupervised_nca + Neural Net	0.765472	32.358003	6.615819
nca + Naive Bayes	0.758596	37.627324	0.016932
Log_bins + Decision Tree	0.754976	0.048297	0.070554
Equal_bins + Decision Tree	0.754253	0.046602	0.071918
nca + Decision Tree	0.732175	39.610076	0.084806
Log_bins + LDA	0.728194	0.048704	0.030460
Equal_bins + LDA	0.712993	0.048417	0.026213
pca + Naive Bayes	0.698516	0.651955	0.015999
nca + LDA	0.695259	39.715770	0.028776
Random_proj + Linear SVM	0.648208	0.042751	2.248036
unsupervised_nca + Linear SVM	0.647123	49.940219	1.275085
unsupervised_lda + Nearest Neighbors	0.642418	9.871763	0.703986
Equal_bins + Naive Bayes	0.642056	0.047311	0.015934
Log_bins + Naive Bayes	0.633008	0.046246	0.016165
pca + LDA	0.628303	0.640071	0.026558
unsupervised_lda + Neural Net	0.623959	9.753802	7.369831
lda + AdaBoost	0.601520	3.736366	0.827525
Random_proj + LDA	0.547593	0.040438	0.027572
unsupervised_nca + Random Forest	0.537097	39.213911	0.105148
Random_proj + Naive Bayes	0.535650	0.041588	0.015931
Random_proj + Random Forest	0.524792	0.039861	0.109490
unsupervised_nca + AdaBoost	0.522620	32.272259	1.317834
unsupervised_lda + Linear SVM	0.519001	10.250297	1.486499
pca + RBF SVM	0.497647	0.648322	6.236839
unsupervised_lda + Random Forest	0.492581	9.547104	0.104321
unsupervised_lda + Naive Bayes	0.467608	9.940516	0.020001
Random_proj + Decision Tree	0.443721	0.039140	0.076492
nca + RBF SVM	0.435396	37.886723	6.847140
unsupervised_nca + Naive Bayes	0.432501	35.660865	0.022299
nca + AdaBoost	0.420919	47.741094	1.639450
unsupervised_nca + Decision Tree	0.419110	33.693980	0.083821
unsupervised_nca + LDA	0.415128	36.129439	0.041690
unsupervised_lda + AdaBoost	0.408976	10.440861	1.212125
lda + RBF SVM	0.405356	3.592771	6.381199
unsupervised_lda + Decision Tree	0.402823	9.988050	0.081020
unsupervised_lda + LDA	0.398118	10.983628	0.061376
Log_bins + AdaBoost	0.364821	0.058048	1.215435
pca + AdaBoost	0.329352	0.641005	1.199264
Random_proj + AdaBoost	0.320666	0.039890	1.298911
Equal_bins + AdaBoost	0.296779	0.049584	1.200615
unsupervised_lda + RBF SVM	0.268549	9.386340	6.468574
unsupervised_nca + RBF SVM	0.263844	51.088881	7.541760
Random_proj + RBF SVM	0.226203	0.040463	5.927608

V. DISCUSSION

In the CWRU dataset, the prevision accuracies are essentially excellent, which confirms similar results from other studies. This dataset is well known for not being extremely challenging in that aspect.

Essentially the simplest projections of the spectra (i.e LDA or even PCA) are sufficient to yield excellent

TABLE II
HUMAN ACTIVITY DATASET

	accuracy	fitting_time	classify_time
pca + Linear SVM	0.826168	0.023069	0.036477
lda + Linear SVM	0.814953	0.091585	0.029380
lda + LDA	0.813084	0.081265	0.003791
nca + Linear SVM	0.813084	15.501449	1.804107
nca + LDA	0.803738	3.125368	0.005575
lda + Nearest Neighbors	0.798131	0.095114	0.076218
pca + LDA	0.796262	0.025189	0.008326
Random_proj + Linear SVM	0.786916	0.002185	0.035992
lda + Naive Bayes	0.786916	0.063292	0.005133
lda + Random Forest	0.783178	0.087501	0.053684
lda + Neural Net	0.781308	0.089556	1.818095
pca + Random Forest	0.773832	0.024143	0.037726
nca + Nearest Neighbors	0.773832	5.976660	0.049898
unsupervised_nca + Linear SVM	0.771963	2.824079	0.036717
Random_proj + LDA	0.771963	0.004330	0.011187
unsupervised_nca + LDA	0.770093	12.745120	0.005673
unsupervised_lda + Nearest Neighbors	0.764486	0.468826	0.076611
unsupervised_lda + LDA	0.762617	0.472181	0.008199
pca + Neural Net	0.758879	0.025533	1.260485
unsupervised_lda + Linear SVM	0.758879	0.415259	0.047638
nca + Naive Bayes	0.758879	8.179425	0.004454
pca + Naive Bayes	0.757009	0.024173	0.004360
unsupervised_nca + Naive Bayes	0.757009	14.459564	0.004721
lda + Decision Tree	0.753271	0.076986	0.008706
nca + Decision Tree	0.749533	5.349274	0.011970
pca + Nearest Neighbors	0.749533	0.030867	0.084109
Random_proj + Nearest Neighbors	0.747664	0.003542	0.099217
nca + Neural Net	0.743925	7.243721	0.895514
unsupervised_nca + Nearest Neighbors	0.740187	5.548086	0.053153
unsupervised_nca + Decision Tree	0.734579	6.388385	0.012035
Random_proj + Decision Tree	0.734579	0.002544	0.011844
unsupervised_lda + Naive Bayes	0.734579	0.430737	0.005741
Random_proj + Random Forest	0.732710	0.002135	0.031100
pca + Decision Tree	0.732710	0.025383	0.015048
nca + Random Forest	0.728972	7.296663	0.030469
unsupervised_nca + Random Forest	0.725234	5.158868	0.030628
Random_proj + Neural Net	0.710280	0.002076	1.509958
unsupervised_lda + Random Forest	0.704673	0.421852	0.035592
unsupervised_nca + Neural Net	0.702804	12.302801	0.466300
unsupervised_lda + Neural Net	0.700935	0.522283	1.761721
Equal_bins + LDA	0.687850	0.004737	0.010364
Random_proj + Naive Bayes	0.687850	0.003202	0.007552
unsupervised_lda + Decision Tree	0.671028	0.444533	0.015734
Log_bins + LDA	0.669159	0.004114	0.006040
Equal_bins + Nearest Neighbors	0.667290	0.007752	0.108640
Log_bins + Nearest Neighbors	0.667290	0.006250	0.062896
Log_bins + Random Forest	0.667290	0.004872	0.028619
Equal_bins + RBF SVM	0.665421	0.007123	0.099206
Log_bins + Naive Bayes	0.661682	0.003965	0.004420
Equal_bins + Neural Net	0.659813	0.005106	4.822545
Log_bins + Decision Tree	0.654206	0.005367	0.002883
Equal_bins + Decision Tree	0.652336	0.004875	0.003691
Log_bins + RBF SVM	0.646729	0.004056	0.084285
Equal_bins + Random Forest	0.646729	0.003805	0.030928
Equal_bins + Naive Bayes	0.642991	0.004129	0.004248
Log_bins + Neural Net	0.629907	0.004023	3.603380
Equal_bins + Linear SVM	0.600000	0.009824	0.080286
Log_bins + Linear SVM	0.596262	0.005201	0.070450
lda + RBF SVM	0.560748	0.095674	0.270099
Log_bins + AdaBoost	0.545794	0.004555	0.161680
lda + AdaBoost	0.542056	0.070310	0.233486
Equal_bins + AdaBoost	0.542056	0.004374	0.187498
Random_proj + AdaBoost	0.527103	0.002332	0.313466
unsupervised_lda + RBF SVM	0.523364	0.430434	0.245666
pca + RBF SVM	0.519626	0.024136	0.249519
Random_proj + RBF SVM	0.508411	0.002088	0.251199
unsupervised_nca + RBF SVM	0.457944	6.717892	0.227130
unsupervised_lda + AdaBoost	0.371963	0.443815	0.306613
pca + AdaBoost	0.371963	0.024460	0.348062
unsupervised_nca + AdaBoost	0.371963	11.069652	0.279476
nca + AdaBoost	0.330841	6.114166	0.284639
nca + RBF SVM	0.190654	5.383328	0.204225

results, when combined with standard classifiers. The figure Fig.3 demonstrates clearly that even a well-chosen 2D projection is almot enough to nicely scatter the dataset and separate the various classes.

NCA is an excellent performer, but is time consuming. The lesson here is that even an extremely simple projector (namely *Equal bins*) is enough to do the job, when combined with an equally simple classifier, for a timing well under 1s. Several studies applied deep learning techniques to such datasets, which is clearly an overkill.

TABLE III
EXCERPT OF RESULTS FOR CASE WESTERN DATASET

	accuracy	fitting_time	classify_time
lda + Linear SVM	1.000000	3.797352	0.037898
lda + Nearest Neigh.	1.000000	3.476086	0.346804
lda + Neural Net	1.000000	3.498505	3.077243
nca + Nearest Neigh.	1.000000	42.344748	0.494811
lda + LDA	0.999638	3.566572	0.024349
nca + Neural Net	0.999638	35.136272	10.746401
lda + Naive Bayes	0.998552	3.461116	0.017265
Equal bins + Near. Neigh.	0.997828	0.059211	0.366956
pca + Nearest Neigh.	0.996381	0.726136	0.354215

TABLE IV
EXCERPT OF HUMAN ACTIVITY DATASET RESULTS

	accuracy	fitting_time	classify_time
pca + Linear SVM	0.826168	0.023069	0.036477
lda + Linear SVM	0.814953	0.091585	0.029380
lda + LDA	0.813084	0.081265	0.003791
nca + Linear SVM	0.813084	15.501449	1.804107
nca + LDA	0.803738	3.125368	0.005575
lda + Nearest Neigh.	0.798131	0.095114	0.076218
pca + LDA	0.796262	0.025189	0.008326

In the case of the HAR dataset, similar results were obtained: essentially the same combination (PCA or LDA followed by linear SVM) obtained the best results (above 80% prediction accuracy). The literature offers better results ([3]) for the accuracy, above 90%, but using much heavier features: an array of more than 500 computed features (various statistical features computed on the raw signals and also spectral features, means of them, deviations, kurtosis, etc ...). In comparison, the pipeline we proposed here is very fast to execute: fast to learn, fast to classify, and light-weight in memory.

VI. Conclusion

Spectral projectors offer a lightweight and robust recipe for creating discriminating features.

The determination of those projectors is fast and easy to tweak. If one desires analog solutions, then the comparatively simpler *band models* we propose offer a handy solution. The more general projectors offer better precision, while still being light-weight. We demonstrated that the features obtained are good enough to provide meaningful results when combined with the most standard classifiers available. Beyond that, one of the main advantages is that the features obtained are easily interpretable. They provide a good candidate for applications in *edge computing*.

References

[1] D. Wang and G. Brown, Eds., *Computational auditory scene analysis: Principles, algorithms and applications*. Wiley-Interscience, 2006.

[2] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome. *The elements of statistical learning. Data mining, inference, and prediction*. Second edition. Springer Series in Statistics. Springer, New York, 2009.

[3] Jorge-L. Reyes-Ortiz, Luca Oneto, Albert Sam , Xavier Parra, Davide Anguita. *Transition-Aware Human Activity Recognition Using Smartphones*. Neurocomputing. Springer 2015.

[4] http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions.

[5] Case Western Reserve University Bearing Data Center Website http://csegroups.case.edu/bearingdatacenter/home

[6] Wade A. Smith, Robert B. Randall, *Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study*, Mechanical Systems and Signal Processing, Volumes 6465, 2015, Pages 100-131.

[7] E. Kim, S. Helal, D. Cook, *Human activity recognition and pattern discovery*.IEEE Pervasive Comput, 2010.