A Whirlwind Survey & Performance Analysis of Supplementing Various Machine Learning Classification Algorithms with KL-Expansions

Final Project Report · MA 751 Spring 2021

Yulin Li, Ali Taqi, Travis Yang, Luke Zheng

05/02/2022

Abstract

Anomaly detection is a field of study important for signal processing. Having well developed anomaly detection methods in limited feature spaces can supplement machine learning classification algorithms, improving performance while improving efficiency. This report attempts to provide more verbose detail to the analytical arguments made by Castrillon-Candas et al to develop a framework for supplementing machine learning classification algorithms using Karhunen-Loeve expansions. This report also extends the framework into a multiclass classification, and assesses the performance of SVM through simulation studies. This report also discusses the potential challenges that may arise from this framework and challenges that we have come across in the making of this report.

1 Introduction

The purpose of this report is to emulate the process of developing the results in Julio E. Castrillon-Candas, Dingning Liu, and Mark Kon's paper which works to create a framework to supplement classification machine learning methods by providing a mechanism by which anomalous signals can be detected using Karhunen-Loeve expansions. Throughout this report, this paper will be what is referred to whenever we say '*the paper*' (italicized for clarity) for the sake of brevity.

The paper is largely built off of another paper written by Castrillon-Candas and Kon which largely discusses the detailed arguments behind the theory used in *the paper*. This paper will be referred to as 'the anomaly detection paper'. The anomaly detection paper provides a lot more detail, though some steps are not fully spelled out in detail that would be beneficial for readers such as us. The anomaly detection paper also provides some added detail to how the results are practically applied in *the paper*. A large focus of this report is to provide detail in a manner that is more accessible to those from other analytical fields and who may have some background in linear algebra and some form of functional analysis, but are not well-versed in parsing or making arguments such as those made in these papers. This may mean expanding the mathematical proofs or manipulations so that the individual steps are clearer, or some arguments/concepts will be compared to things that would be more familiar to the reader or closer to home. There are many parallelisms between the theory of the more commonly known principal component analysis and the theory of Karhunen-Loeve expansions, so there will be a theme of comparing the results when possible.

In addition to expanding and analogizing most of the theoretical arguments needed to develop *the paper*, we will be attempting to re-implement this framework on other datasets. The code will be largely borrowed from the Github repository that Castrillon-Candas provided for us. Adjustments to the method and framework will be made to apply the method in a multiclass classification setting as well.

2 Methods

2.1 Principal Component Analysis

To begin discussing the analytical methods behind the framework proposed in *the paper*, it would be helpful to introduce the concept of principal component analysis (PCA) to build some basic intuition for the notation in the Karhunen-Loeve (KL) expansion.

Suppose we have a collection of data points in p-dimensional space, X is $n \times p$. Without loss of generality, we consider the case when X is centered, in other words E(X) = 0. The idea behind PCA is to provide the best view of X as possible. This is done by creating a new coordinate system of p-dimension. This new coordinate system is based on p principal components (PC) called $(v_1, \ldots, v_p)$ which are effectively an orthonormal basis. These PC's have an ordinality to them as well, such that the variance of the projections of X, Xv_i, onto each PC is in descending order, so the projection onto the first PC has the largest variance and the variance of the pth PC is the smallest.

$$\begin{aligned} \text{Var}(Xv_i) &= v_i^T X^T X v_i = v_i^T \Sigma v_i \\ v_1^T \Sigma v_1 &\geq v_2^T \Sigma v_2 \geq ... \geq v_p^T \Sigma v_p \end{aligned}$$

The intuition behind this process is that it would provide a reasonable method of dimension reduction. By simply taking the first k principal components, one could capture the majority of the variance in the data. This is akin to the common practice of taking a 2D picture of a 3D object. In order to get the best picture, it is important to select the best angle to see the 3D object in as much of its entirety as possible. This is translated mathematically as finding the first two PC's that maximize the variance of the data on those components. Doing this mathematically, we may focus on the first PC, and choose to maximize the variance of the projection. Since we also imposed a constraint on v_1 having unit magnitude (from the fact that the basis of v's is orthonormal), we may use the method of lagrangian multipliers to maximize the variance.

$$\begin{aligned} & v_1^T \Sigma v_1 - \lambda_1 (v_1^T v_1 - 1) \\ 0 = & \frac{\partial}{\partial v_1} v_1^T \Sigma v_1 - \lambda_1 (v_1^T v_1 - 1) \\ 0 = & \Sigma v_1 - \lambda_1 v_1 \\ 0 = & (\Sigma - \lambda_1 I_p) v_1 \end{aligned}$$

When the lagrangian is written in this form, it is clear that the principal component v_1 that maximizes the variance of its projection Xv_1 occurs at an eigenvalue of Σ, λ_1 and the principal component is its corresponding eigenvector. Selection of the eigenvalue needs to maximize the variance though, but we can show that maximizing the eigenvalue is equivalent to maximizing the projection's variance.

$$\begin{aligned} 0 = & \Sigma v_1 - \lambda_1 v_1 \\ 0 = & v_1^T \Sigma v_1 - v_1^T \lambda_1 v_1 \\ v_1^T \Sigma v_1 = & v_1^T \lambda_1 v_1 \\ = & \lambda_1 v_1^T v_1 \\ \text{Var}(Xv_1) = & \lambda_1 \text{ since } v_1^T v_1 = 1 \end{aligned}$$

Since the variance is equal to the eigenvalue, then maximizing the variance equates to finding the maximum eigenvalue and its corresponding eigenvector to obtain the first principal component. Finding subsequent PC's is now a manner of ordering the remaining eigenvalues and finding their corresponding eigenvectors, which must be orthogonal to each other. This allows us to construct a basis of PC's, and using this basis we may re-express the centralized data matrix X onto a new coordinate system as

$$\tilde{X} = \sum_{k=1}^{p} X v_k$$

such that the variance of the data is decreasing along each component. Since the eigenvalues are shown above to directly correspond with the variances of each PC's projection which are orthogonal/independent of each other, we may say that the sum of the eigenvalues is like a measure of the total variance in the data, so the relative proportion of each λ to their sum is a measure of the proportion of total variance explained by each PC.

PCA is a very useful result. Its beauty arises from its sheer simplicity, in that with some very basic linear algebra, it is simple to find the perpendicular components or axes that spread out the data as much as possible. It is also possible to obtain the exact same result by finding the axes that minimize the square of the perpendicular distances from each point to each axis. One would find that the eigenvector that corresponds with the largest eigenvalue would minimize the square of the perpendicular distances from each point at the same time as maximizing the variance of the projections onto that axis. It should be noted that minimizing the square perpendicular distances of every point is distinct from least squares regression since in regression, the minimization is on the square vertical distances along a particular outcome variable, not the perpendicular distances which is in a sense symmetric against all variables.

2.2 Karhunen-Loeve Expansions

To begin talking about KL expansions, we first briefly introduce the concept of eigenfunctions. Eigenfunctions are mechanically the same as eigenvectors, but the vectors in the space of interest are functions. However due to the nature of function spaces, these eigenfunctions may also be infinite dimensional. In a function space with a linear operator T, its nontrivial eigenvalues L and eigenfunctions f satisfy the equation $Tf = \lambda f$. As one would expect for such an analogous concept, properties of eigenvectors carry over to eigenfunctions, particularly the orthogonality of a linear operator's eigenfunctions.

To set up the scenario for a KL expansion, consider a stochastic process $v(x)$ for $x \in X$ with $E(v) = 0$ has a kernel function $K(x, y) = \text{Cov}(v(x), v(y))$ which maps from $X \times X \to \mathbb{R}$. The goal is to represent find an orthogonal basis $\phi_k(x)$ and independent random variables r_k with assumed mean of 0 and unit variance such that

$$v(x) = \sum_{k=1}^{\infty} \sqrt{\lambda_k} r_k \phi_k(x)$$
$$r_k = \frac{1}{\sqrt{\lambda_k}} \int_X v(x)\phi_k(x)dx$$

To do so, consider a linear operator T as below and compute its eigenvalues and eigenfunctions

$$T(f)(y) = \int_X K(x, y)f(x)dx$$
$$\int_X K(x, y)\phi_k(x)dx = \lambda_k \phi_k(y)$$

The sequence of eigenvalues also satisfies a property similar to that introduced in PCA in the previous section where $\lambda_1 \geq \lambda_2 \geq \cdots > 0$.

Originally $v(x)$ was said to have expectation 0 for simplicity, in the case of non-zero expectation, then its expectation can easily be subtracted and added back on in the expansion form.

In Castrillon-Candas et al, the scenario is very similar, though it is equipped with more detail to make it applicable to more complex topologies. There is a probability space (Ω, F, P), X is some domain of real numbers, and $L^2(X)$ is a Hilbert space that maps $X ß R$ with standard inner product. There is also a space of random variables $L^2_P(\Omega; L^2(X))$ that maps from $\Omega ß L^2(X)$ with the below inner product.

$$\langle u, v \rangle L^2_{\mathbb{P}}(\Omega; L^2(X)) = \int \Omega \langle u, v \rangle_{L^2(X)} d\omega$$
$$\langle u, v \rangle_{L^2(X)} = \int_X uv dx$$

The KL expansion becomes a KL tensor product expansion with the following appearance for a process that may not have expectation 0

$$v_M(x,\omega) = E_v + \sum_{k=1}^{M} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega)$$

We may easily provide detail to evaluate the norm of the expansion below

$$\begin{aligned}
||v - E_v||^2_{L^2_{\mathbb{P}}(\Omega;L^2(X))} &= \int_\Omega \int_X \sum_{j=1}^{\infty} \lambda_j^{\frac{1}{2}} \phi_k(x) r_j(\omega) \sum_{k=1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) dx d\mathbb{P} \\
&= \int_\Omega \int_X \sum_{j \neq k} \lambda_j^{\frac{1}{2}} \lambda_k^{\frac{1}{2}} \phi_k(x) \phi_k(x) r_j(\omega) r_k(\omega) \\
&\quad + \sum_{j=k} \lambda_j^{\frac{1}{2}} \lambda_k^{\frac{1}{2}} \phi_k(x) \phi_k(x) r_j(\omega) r_k(\omega) dx d\mathbb{P} \\
&= \int_\Omega \int_X \sum_{j \neq k} \lambda_j^{\frac{1}{2}} \lambda_k^{\frac{1}{2}} 0 dx d\mathbb{P} \\
&\quad + \int_\Omega \int_X \sum_{k=1}^{\infty} \lambda_k \phi_k^2(x) r_k^2(\omega) dx d\mathbb{P} \\
&= \sum_{k=1}^{\infty} \lambda_k \int_\Omega r_k^2(\omega) \int_X \phi_k^2(x) dx d\mathbb{P} \\
&= \sum_{k=1}^{\infty} \lambda_k \int_\Omega r_k^2(\omega) 1 d\mathbb{P} \\
&= \sum_{k=1}^{\infty} \lambda_k
\end{aligned}$$

Which leverages the fact that $\phi_k(x)$ are orthonormal and the random variables r_k are independent with zero mean and unit variance. This representation of the norm of the centered expansion is like a measure of the total amount of variance that can be accounted for in the stochastic process of interest. It is easy to compare this to the interpretation of the sum of all eigenvalues in PCA, which corresponds with the sum of variances accounted for by each PC. It should be noted that we believe there is a typo in *the paper*'s text on page 3 where this result is shown, where the sum of $\lambda_k^{\frac{1}{2}}$ was written when it should just be the sum of λ_k. This is correctly written in the anomaly detection paper though.

Since in practice it is not feasible to obtain infinite eigenvalue and eigenfunction estimates, a pragmatic cutoff value M needs to be selected for which to stop the eigenvalue/eigenfunction estimation. Ideally, higher M should result in lower error in the approximation. This results in a truncated KL expansion in the form of a partial sum.

$$v_M(x,\omega) = E_v + \sum_{k=1}^{M} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega)$$

It is easy to show that the residual of this approximation is of the following form

$$
\begin{aligned}
||v - v_M||^2_{L^2_{\mathbb{P}}(\Omega;L^2(X))} &= \int_\Omega \int_X \sum_{j=1}^{\infty} \lambda_j^{\frac{1}{2}} \phi_k(x) r_j(\omega) \sum_{k=1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) \\
&\quad - \sum_{j=1}^{M} \lambda_j^{\frac{1}{2}} \phi_k(x) r_j(\omega) \sum_{k=1}^{M} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) dx d\mathbb{P} \\
&= \int_\Omega \int_X \sum_{j=1}^{\infty} \lambda_j^{\frac{1}{2}} \phi_k(x) r_j(\omega) \sum_{k=1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) dx d\mathbb{P} \\
&\quad - \int_\Omega \int_X \sum_{j=1}^{M} \lambda_j^{\frac{1}{2}} \phi_k(x) r_j(\omega) \sum_{k=1}^{M} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) dx d\mathbb{P} \\
&= ||v - E_v||^2_{L^2_{\mathbb{P}}(\Omega;L^2(X))} - ||v_M - E_v||^2_{L^2_{\mathbb{P}}(\Omega;L^2(X))} \\
&= \sum_{k=1}^{\infty} \lambda_k - \sum_{k=1}^{M} \lambda_k \\
&= \sum_{k=M+1}^{\infty} \lambda_k
\end{aligned}
$$

Just by looking at this quantity comparing the partial sum with the infinite sum, it is clear that selecting larger M for the partial sum naturally leads to better approximations. This is directly analogous to the interpretation of the λ_k from PCA where their sum is interpreted as the contribution of variance for each component of this expansion. The quantity above effectively represents the variance that's leftover and is not accounted for in the approximation from the truncated KL expansion.

As it turns out, it can be shown that this residual quantity is an infimum over all possible such expansions, meaning that this approximation by way of the truncated KL expansion is the best approximation possible for a given M. This result is shown in theorem 2.7 by Schwab et al.

The selection of such an M is effectively an engineering problem of balancing desired accuracy against the computational cost of estimating eigenvalues and eigenfunctions. Such a selection could be analogized to the case of PCA where it may be of interest to select only the first few PC's for the purposes of data visualization. The selection of this M may be driven by cross-validation during practical implementation.

2.3 Application of KL Expansions

The overall purpose of developing this theory for representing stochastic processes is to apply it to data to improve classification using machine learning methods. To do so, *the paper* constructs a scenario where some data has been expended to estimate the truncated KL expansion, thereby estimating an eigenspace, a space spanned by the eigenfunctions from the truncated KL expansion. This eigenspace would be estimated by training data from a single class, this class is a sort of reference class. The intuition behind this is that this eigenspace would be specific to the reference class that it was trained on. The motivation for this representation is that the reference class data would generally take values within the estimated eigenspace, and data from other classes may take on values within the same estimated eigenspace, however some data would presumably lay outside of the estimated eigenspace. These anomalous data points would be used to help classify data against the reference class.

To represent the intuition mathematically, *the paper* presents the estimated eigenspace as the smallest of a sequence of nested subspaces. The closure of the union of this sequence of subspaces is the function space of interest, $L^2(X)$.

$$\begin{aligned} V_0 =& \text{span}(\phi_1, ..., \phi_M) \\ V_0 \subset& V_1 \subset ... \subset L^2(X) \\ & \overline{\bigcup_{k \in N_0} V_k} = L^2(X) \end{aligned}$$

Each subspace V_{k+1} is iteratively constructed by taking the direct sum of the previous subspace V_k and an orthogonal subspace W_k.

$$\begin{aligned} & W_k \perp V_0 \forall k \in N_0 \\ & V_{k+1} = V_k \oplus W_k \\ & \overline{V_0 \oplus W_0 \oplus W_1 \oplus ... \oplus W_k} = L^2(X) \end{aligned}$$

An elementary example that helped us visualize and understand this scenario is if we considered the eigenspace V_0 to be something like R^2 and instead of $L^2(X)$ we were concerned with something like R^5, then one could imagine that W_0, W_1, and W_2 would each be the third, fourth, and fifth real components of R^5 respectively, which are all orthogonal to each other and to V_0. Clearly their direct summation of all of those spaces would amount to R^5 the space of interest.

With this representation, we have a framework to assess the magnitude of a potentially anomalous signal that partially resides outside of the estimated eigenspace V_0. Recall that the eigenspace V_0 would have been trained on a class which we are using as a reference to compare other classes against. Hence, it's intuitive that larger magnitudes from the W_k components indicate evidence against the data point being in the reference class.

To further help quantify anomalies, *the paper* presents a lemma where we have a signal $v(x, \omega)$ with the full KL expansion defined previously. With this KL expansion, we may assess the behavior of its projection onto the W_k components orthogonal to the estimated eigenspace. The projections are presented below as $d_k^l(\omega)$ where $l \in N_0$ and $k = 1, ..., M_l$ where M_l is the dimension of each subspace W_l. The indices are somewhat confusing in *the paper* and requires some head-scratching to keep track of since previously k was used to index the orthogonal subspaces as W_k, but now it is used to index a component of the W_l subspace since these orthogonal subspaces may be more than one dimension, and each one does not necessarily have the same dimension. Hence one could say that the projection d_k^l is like the kth component of the lth orthogonal subspace component. Each of these W_l orthogonal subspaces are also equipped with an orthonormal basis $(\psi_1^l, ..., \psi_k^l, \ldots, \psi_{M_l}^l)$. These projection coefficients are later referred to as the multilevel coefficients, or coefficients from multiple levels of resolution. The levels of resolution correspond with the l index, in other words higher levels of l correspond with higher levels of resolution of the orthogonal space.

$$
\begin{aligned}
d_k^l(\omega) = \langle v(x,\omega), \psi_k^l \rangle_{L^2(X)} &= \int_X v(x,\omega)\psi_k^l(x)dx \\
&= \int_X \sum_{j=1}^{\infty} \lambda_j^{\frac{1}{2}} \phi_j(x) r_j(\omega) \psi_k^l(x) dx \\
&= \sum_{j=1}^{\infty} r_j(\omega) \int_X \lambda_j^{\frac{1}{2}} \phi_j(x) \psi_k^l(x) dx \\
E[d_k^l(\omega)] &= \int_\Omega d_k^l(\omega) d\omega \\
&= \int_\Omega \sum_{j=1}^{\infty} r_j(\omega) \int_X \lambda_j^{\frac{1}{2}} \phi_j(x) \psi_k^l(x) dx d\omega \\
&= \sum_{j=1}^{\infty} \int_X \lambda_j^{\frac{1}{2}} \phi_j(x) \psi_k^l(x) dx \int_\Omega r_j(\omega) d\omega \\
&= \sum_{j=1}^{\infty} \int_X \lambda_j^{\frac{1}{2}} \phi_j(x) \psi_k^l(x) dx E[r_j(\omega)] \\
&= \sum_{j=1}^{\infty} \int_X \lambda_j^{\frac{1}{2}} \phi_j(x) \psi_k^l(x) dx (0) \\
&= 0
\end{aligned}
$$

This conclusion about the expectation of the projections is easy to show using the fact that $E[r_j(\omega)] = 0$ from earlier assumptions. Below we will also use the assumption that $E[r_j(\omega)^2] = 1$ and $E[r_i(\omega) r_j(\omega)] = 0 \; \forall i \neq j$.

Another quantity of interest is the expected value of the squared projections, which are effectively the variances of the projections since the projections are shown to have zero expectation.

$$
\begin{aligned}
E[(d_k^l)^2] &= \int_\Omega (d_k^l(\omega))^2 d\omega \\
&= \int_\Omega \sum_{i,j} r_i(\omega) r_j(\omega) \int_X \lambda_i^{\frac{1}{2}} \phi_i(x) \psi_k^l(x) dx \int_X \lambda_j^{\frac{1}{2}} \phi_j(x) \psi_k^l(x) dx d\omega \\
&= \sum_{i,j} \int_X \lambda_i^{\frac{1}{2}} \phi_i(x) \psi_k^l(x) dx \int_X \lambda_j^{\frac{1}{2}} \phi_j(x) \psi_k^l(x) dx \int_\Omega r_i(\omega) r_j(\omega) d\omega \\
&= \sum_{i \neq j} \int_X \lambda_i^{\frac{1}{2}} \phi_i(x) \psi_k^l(x) dx \int_X \lambda_j^{\frac{1}{2}} \phi_j(x) \psi_k^l(x) dx E[r_i r_j] \\
&\quad + \sum_{i=j} \int_X \lambda_i^{\frac{1}{2}} \phi_i(x) \psi_k^l(x) dx \int_X \lambda_j^{\frac{1}{2}} \phi_j(x) \psi_k^l(x) dx E[r_i r_j] \\
&= \sum_{i \neq j} \int_X \lambda_i^{\frac{1}{2}} \phi_i(x) \psi_k^l(x) dx \int_X \lambda_j^{\frac{1}{2}} \phi_j(x) \psi_k^l(x) dx (0) \\
&\quad + \sum_{i=j} \int_X \lambda_i^{\frac{1}{2}} \phi_i(x) \psi_k^l(x) dx \int_X \lambda_j^{\frac{1}{2}} \phi_j(x) \psi_k^l(x) dx (1) \\
&= \sum_{j=1}^{\infty} \lambda_j \left(\int_X \phi_j(x) \psi_k^l(x) dx \right)^2 \\
&= \sum_{j=1}^{M} \lambda_j \left(\int_X \phi_j(x) \psi_k^l(x) dx \right)^2 \\
&\quad + \sum_{j=M+1}^{\infty} \lambda_j \left(\int_X \phi_j(x) \psi_k^l(x) dx \right)^2 \\
&= \sum_{j=1}^{M} \lambda_j \langle \phi_j(x), \psi_k^l(x) \rangle^2 + \sum_{j=M+1}^{\infty} \lambda_j \langle \phi_j(x), \psi_k^l(x) \rangle^2
\end{aligned}
$$

The first quantity in the last line may be reduced to 0 because we established that any ψ_k^l is a component of a basis for the subspace W_l which is orthogonal to V_0, which is the subspace that is spanned by the basis $\{\phi_j\}_{j \leq M}$. Hence, since ψ_k^l is an element of W_l, and ϕ_j is an element of V_0, and by construction $W_1 \perp V_0$, then $\langle \phi_j(x), \psi_k^l(x) \rangle = 0 \; \forall i \leq M$.

$$
\begin{aligned}
E[(d_k^l)^2] &= \sum_{j=1}^{M} \lambda_j (0)^2 + \sum_{j=M+1}^{\infty} \lambda_j \langle \phi_j(x), \psi_k^l(x) \rangle^2 \\
&= \sum_{j=M+1}^{\infty} \lambda_j \langle \phi_j(x), \psi_k^l(x) \rangle^2
\end{aligned}
$$

From here, the Cauchy-Schwarz inequality may be used to develop an upper bound for the quantity, along with the fact that the bases ϕ and ψ are orthonormal.

$$\begin{aligned}
\langle x, y\rangle \leq & \sqrt{\langle x, x\rangle \langle y, y\rangle} \\
E[(d_k^l)^2] = & \sum_{j=M+1}^{\infty} \lambda_j \langle \phi_j(x), \psi_k^l(x)\rangle^2 \\
\leq & \sum_{j=M+1}^{\infty} \lambda_j \langle \phi_j(x), \phi_j(x)\rangle \langle \psi_k^l(x), \psi_k^l(x)\rangle \\
\leq & \sum_{j=M+1}^{\infty} \lambda_j (1)(1) \\
\leq & \sum_{j=M+1}^{\infty} \lambda_j
\end{aligned}$$

In other words, the expected variance for each component projection in the orthogonal subspaces W_l are controlled by the sum of the eigenvalues not included in the truncated KL expansion. Then using Chebyshev's inequality, the following relationship may be constructed

$$\begin{aligned}
P(|X - EX| \geq c) \leq & \frac{\sigma^2}{c^2} \\
P(|d_k^l - 0| \geq c) \leq & \frac{\sum_{j=M+1}^{\infty} \lambda_j}{c^2} \\
P(|d_k^l| \geq c) \leq & \frac{\sum_{j=M+1}^{\infty} \lambda_j}{c^2}
\end{aligned}$$

With this relationship, it is clear that if $\sum_{j=M+1}^{\infty} \lambda_j$ goes to 0 as M goes to infinity, then the right side goes to 0 as M goes to infinity, which means that the magnitude of the projection converges in probability to 0. In other words, since M controls the variance of d_k^l, we have great certainty about the magnitude of d_k^l for very large M, assuming the observed signal is actually a realization of the stochastic process of interest $v(x, \omega)$. In the event that the signal is anomalous and perhaps from a different process, then we may observe larger values of d_k^l.

The paper then presents a scenario where a signal $u(x, \omega)$ can be decomposed into the finite-dimensional truncated KL expansion which resides in the eigenspace of V_0, and the components in the orthogonal subspaces W_l. As a reminder, we are working in the case where the signal has 0 expectation.

$$\begin{aligned}
u(x, \omega) = & v_M(x, \omega) + w(x, \omega) \\
w(x, \omega) \in & L^2_{\mathbb{P}}(\Omega; L^2(X))
\end{aligned}$$

This can be seen as a sort of decomposition into a vector's perpendicular components, such as in a simplistic two-dimensional case where a vector on an xy-plane can be represented as the sum of its x component and y component. It can be easily shown using many of the techniques in the intermediary steps in previous derivations and the properties of inner products that

$$\begin{aligned}
||w(x, \omega)||^2_{L^2(X)} = & \sum_{l \in N_0} \sum_{k=1}^{M_l} (d_k^l)^2 \\
||w(x, \omega)||^2_{L^2_{\mathbb{P}}(\Omega; L^2(X))} = & \sum_{l \in N_0} \sum_{k=1}^{M_l} E[(d_k^l)^2]
\end{aligned}$$

To add on to this scenario, *the paper* considers when a signal may require the infinite dimensional KL expansion to be correctly represented. This is the same as the previous representation of $u(x,\omega)$, except the remaining $M+1$ to ∞ terms of the KL expansion were hidden in the $w(x,\omega)$ previously. We will also consider the perpendicular space $V_0^\perp$ which is the span of the bases $\{\{\phi_k^l\}_{k=1}^{M_l}\}_{l\in N_0}$, or the direct sum of the orthogonal subspaces W_l. We can consider the orthogonal projection from the larger space $L^2(X)$ onto $V_0^\perp$ as $P: L^2(X) \to V_0^\perp$ to get the components of the KL expansion and the $w(x,\omega)$ component that are orthogonal to the truncated KL expansion which resides in the space V_0.

$$\begin{aligned} u(x,\omega) =& v(x,\omega) + w(x,\omega) \\ =& v_M(x,\omega) + \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) + w(x,\omega) \\ Pu(x,\omega) =& \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) + w(x,\omega) \end{aligned}$$

The eventual goal of this statement is to establish an interval around $\sum_{l\in N_0}\sum_{k=1}^{M_l} E[(d_k^l)^2]$. This can be done by equating the $w(x,\omega)$ from earlier to $Pu(x,\omega)$, as they are corresponding components of their respective equations for $u(x,\omega)$. The proof outlined in *the paper*, while detailed, was one of the more confusing ones to parse due to its multiple uses of the Cauchy-Schwarz inequality. Hopefully the following rephrasing of the proof is more verbose for the reader albeit more long-winded.

$$\begin{aligned} \text{Let } t_M =& || \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) ||^2_{L^2_{\mathbb{P}}(\Omega; L^2(X))} \\ \sum_{l\in N_0}\sum_{k=1}^{M_l} E[(d_k^l)^2] =& ||w(x,\omega)||^2_{L^2_{\mathbb{P}}(\Omega;L^2(X))} = ||Pu(x,\omega)||^2_{L^2_{\mathbb{P}}(\Omega;L^2(X))} \\ =& || \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) + w(x,\omega) ||^2_{L^2_{\mathbb{P}}(\Omega;L^2(X))} \\ =& \Bigg\langle \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) + w(x,\omega), \\ & \qquad \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) + w(x,\omega) \Bigg\rangle_{L^2_{\mathbb{P}}(\Omega;L^2(X))} \\ =& \Bigg\langle \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega), \\ & \qquad \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) \Bigg\rangle_{L^2_{\mathbb{P}}(\Omega;L^2(X))} \\ & + \langle w(x,\omega), w(x,\omega) \rangle_{L^2_{\mathbb{P}}(\Omega;L^2(X))} \\ & + 2 \Bigg\langle \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega), w(x,\omega) \Bigg\rangle_{L^2_{\mathbb{P}}(\Omega;L^2(X))} \\ =& || \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} \phi_k(x) r_k(\omega) ||^2_{L^2_{\mathbb{P}}(\Omega;L^2(X))} \\ & + ||w(x,\omega)||^2_{L^2_{\mathbb{P}}(\Omega;L^2(X))} \\ (\star) \qquad & + 2 \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} E[\langle \phi_k(x) r_k(\omega), w(x,\omega) \rangle_{L^2(X)}] \end{aligned}$$

By applying the Cauchy-Schwarz inequality for the probabilistic measure on the expected value in $(\star)$, we may obtain the following inequality for it.

$$\begin{aligned}\left|E[\langle \phi_k(x) r_k(\omega), w(x,\omega)\rangle]\right| &\leq \int_X ||\phi_k(x) r_k(\omega)||_{L^2_{\mathbb{P}}(\Omega)} ||w(x,\omega)||_{L^2_{\mathbb{P}}(\Omega)} dx \\ &\leq \int_X |\phi_k(x)| \; ||w(x,\omega)||_{L^2_{\mathbb{P}}(\Omega)} dx\end{aligned}$$

Then by applying the Cauchy-Schwartz inequality again on the Lebesgue measure.

$$\begin{aligned}\left|E[\langle \phi_k(x) r_k(\omega), w(x,\omega)\rangle]\right| \leq& \int_X |\phi_k(x)| \; ||w(x,\omega)||_{L^2_{\mathbb{P}}(\Omega)} dx \\ \leq& ||\phi_k(x)||_{L^2(X)} \; ||w(x,\omega)||_{L^2_{\mathbb{P}}(\Omega; L^2(X))} \\ \leq& ||w(x,\omega)||_{L^2_{\mathbb{P}}(\Omega; L^2(X))}\end{aligned}$$

The above result uses the fact that the eigenfunctions ϕ_k are orthonormal. Now that we have the bounds for the absolute value for the expected value, we may generate an interval for the expected value itself. Then by plugging in each of these inequalities for the expected value one at a time into the $(\star)$ above.

$$\begin{aligned}\left|E[\langle \phi_k(x) r_k(\omega), w(x,\omega)\rangle]\right| \leq& ||w(x,\omega)||_{L^2_{\mathbb{P}}(\Omega; L^2(X))} \\ -||w(x,\omega)||_{L^2_{\mathbb{P}}(\Omega; L^2(X))} \leq& \quad E[...] \quad \leq ||w(x,\omega)||_{L^2_{\mathbb{P}}(\Omega; L^2(X))}\end{aligned}$$

Then by plugging in each of these inequalities for the expected value one at a time into the $(\star)$ above.

$$\begin{aligned}\sum_{l \in N_0} \sum_{k=1}^{M_l} E[(d_k^l)^2] \leq& t_M + ||w(x,\omega)||^2_{L^2_{\mathbb{P}}(\Omega; L^2(X))} \\ &+ 2 \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} ||w(x,\omega)||_{L^2_{\mathbb{P}}(\Omega; L^2(X))}\end{aligned}$$

and

$$\begin{aligned}\sum_{l \in N_0} \sum_{k=1}^{M_l} E[(d_k^l)^2] \geq& t_M + ||w(x,\omega)||^2_{L^2_{\mathbb{P}}(\Omega; L^2(X))} \\ &- 2 \sum_{k=M+1}^{\infty} \lambda_k^{\frac{1}{2}} ||w(x,\omega)||_{L^2_{\mathbb{P}}(\Omega; L^2(X))}\end{aligned}$$

The result we show here is not as elegant as the one presented in *the paper* though. We believe that there was some typo involved with a missing square root, which prevents the last term in each of the above inequalities being simplified to just t_M as is done in *the paper* and the anomaly detection paper. A similar typo was made in an earlier portion of *the paper* on the top of page 3 where the sum of $\lambda_k^{\frac{1}{2}}$ was mistyped when it was likely supposed to be the sum of λ_k as mentioned earlier in this section. This interval was not yet implemented into the code though, so it was never directly applied in the examples that *the paper* or that we work with.

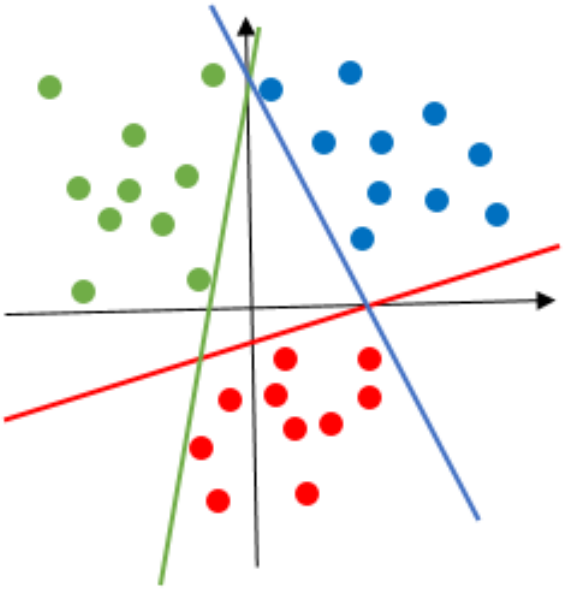
2.4 Support Vector Machines

The main machine learning method that is focused on in *the paper* are support vector machines (SVM), so we will provide a brief overview of the method as well.

SVM is very similar to methods like linear discriminant analysis (LDA) in that it generates some hyperplane in the data input space that provides a rule for classification. In the case of LDA, there needs to be a boundary for which the data can be perfectly separated by class, a sort of hard boundary. However, SVM offers some flexibility for data that are inseparable, providing a soft boundary for which to classify data by.

Cortes and Vapnik also present a method of convolution of a dot product in a feature space. This is effectively a way to implement some kernel method to SVM. If the data are not easily separable in the original input data space, then it is possible to transform the data into a higher dimensional space to make them more linearly separable. While it is always possible to map into a larger feature space to improve the linear separability of the data, its implementation needs to be done with care, since poor selection of the kernel and the kernel's parameters may result in overfit for limited data.

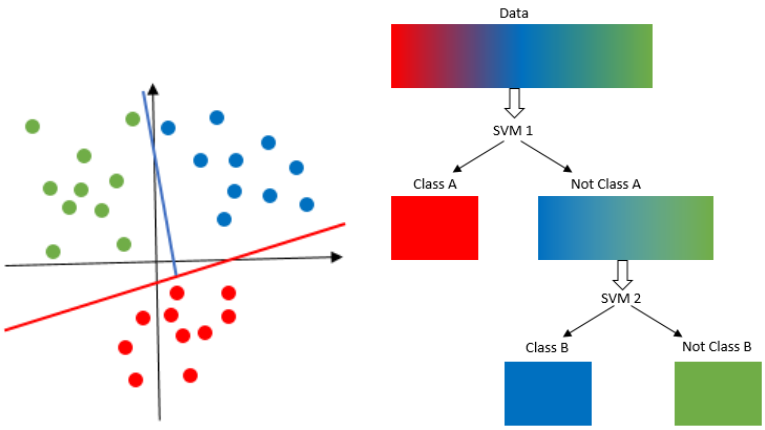
SVM can also be used for multiclass classification, though the commonly used methods appear somewhat contrived. Methods such as one-versus-one or one-versus-all SVMs seem to dominate multiclass classification, though each of those require the training of many SVMs. Both methods are just collections of binary classification SVMs, and some aggregate of the decisions by each binary classifier is used to determine the overall algorithm's decision. The one-versus-one SVM is much less efficient, requiring $\binom{k}{2}$ SVMs since an SVM is created for every pairwise comparison between the k classes. One-versus-all is slightly more efficient, only requiring k SVMs since it is an aggregate of $\binom{k}{1}$ comparisons between each class and all other classes together, in other words, each class is compared against the $k-1$ other classes for each binary classifier. Below is a representation of what a one-versus-rest SVM may appear as for classifying three groups. Each colored line or hyperplane represents the SVM used to classify between its color versus the other two colors together.



There are also structured SVMs which support multiclass classification, though their implementations do not appear as readily available. They also may not necessarily perform much better than the established one-versus-one and one-versus-rest methods.

2.5 Greedy Tree Multiclass SVM

We also propose a greedy tree nested sequence of binary SVMs similar to a one-versus-rest approach to multiclass classification, though our approach would aim to have $k-1$ SVMs instead of k SVMs. To start an SVM would be trained to differentiate between some class and the rest of the classes combined, just like a one-versus-rest. This would be repeated for the next class, except the second SVM would only be trained against data not already classified into the first class. Hence this SVM is sort of nested within the previous SVM. Below is a visual representation of this method in a three-class scenario. Its generalization to k-classes should be clear, since it can be imagined that the third class (green in the example below) can be further subdivided into other classes.



The construction of this set of nested SVMs may potentially lend itself to being biased towards the first classes being classified in the nested structure of SVMs, since once a data point has been classified as the first class (red in the above example) in the first SVM, then it has no chance of being considered for other classes in the subsequent nested SVMs. Because of this, we may expect to see asymmetric performance, in the sense that the selection of the class being classified by the first SVM will affect the method's performance which may not be a desirable property of such a classifier.

Perhaps this application then would be more suited for data where it is common to see large imbalances in the distribution of classes of the data. In which case, the choice of which class to classify first for each subsequent SVM would be simply the order of the commonality of each class. For example, the most common class would be selected for the first SVM, the second most common for the second, and so on. We hope to do some simulation studies to test the performance of this framework against the more traditional one-versus-rest implementation.

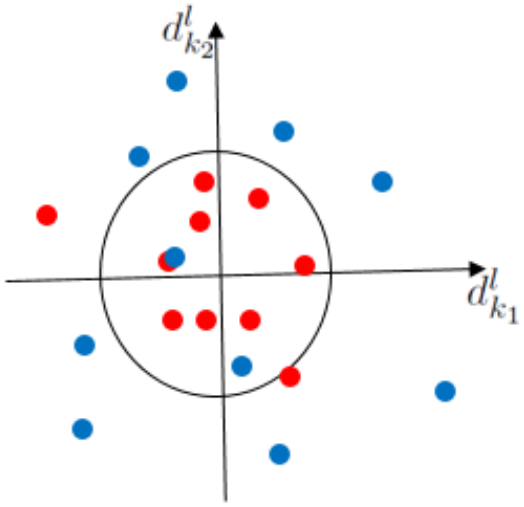
2.6 Impelmentation Algorithm

Now we will outline the general approach described in *the paper* for implementing the KL expansions and the orthogonal subspace projection coefficients (also referred to in *the paper* as the multilevel coefficients from the multilevel filter) to supplement a SVM.

The idea is that for a given training set of data, some data needs to be allocated to train the multilevel filter for estimating the eigenfunctions and multilevel coefficients, and the rest will be allocated to train the SVM. However, only data from a single class would be used to train the multilevel filter, because the goal is to represent the eigenspace for a particular process, or in this case, for a particular class, a sort of reference class to classify against. The goal would be to look for data that have multilevel projection coefficients that lay outside of the expected boundary based on the interval for the variance of the coefficients derived previously, and those would be considered anomalous points, or points that are not representative of the reference class, suggesting that they actually belong to the other class in a binary classification problem.

Below is a figure heavily inspired by Figure 1 in *the paper* which provides a very useful visualization for how a the multilevel coefficients would manifest for the reference class (red) that was used for estimating

the multilevel filter, and how the coefficients would manifest for the alternative class of interest (blue) for a simple binary classification problem.

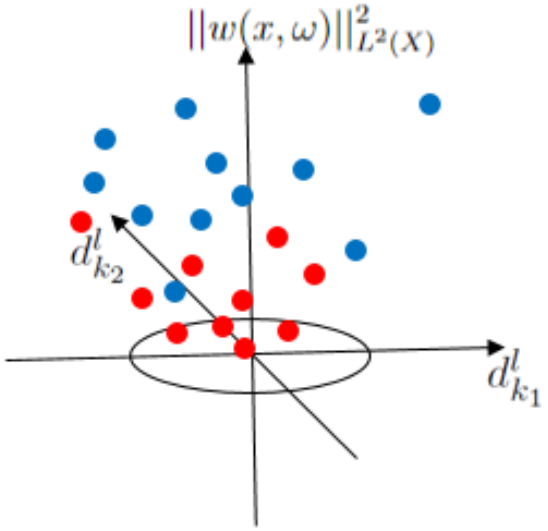


Clearly those of the reference class that was used to build the multilevel filter would have smaller coefficients, and those from outside the reference class would have larger coefficients. To connect this back to the section on application of KL expansions, looking at the components of a decomposed signal

$$\langle u, v \rangle_{L^2_{\mathbb{P}}(\Omega; L^2(X))} = \int_\Omega \langle u, v \rangle_{L^2(X)} d\omega$$
$$\langle u, v \rangle_{L^2(X)} = \int_X uv dx$$

We would consider $w(x, \omega)$ to be small if the signal was truly a manifestation of the reference class, and it would be large if the signal was not a manifestation of the reference class. The size of $w(x, \omega)$ has been shown to correspond to the magnitudes of the projection coefficients, so this provides a convenient method of differentiating between classes. These projection coefficients are fed into a SVM to give it more useful features to classify the data with. The SVM will use a radial basis function kernel to transform the data into a higher dimensional space so that it is more easily linearly separable by a simple hyperplane. The motivation for this is easy to show visually by considering the previous figure which is a simplification of this method down to an easy to grasp two-dimensional scenario, where only two multilevel projection coefficients are of interest. Clearly it would be difficult to draw a one-dimensional hyperplane, or a line, to provide a reasonable classifier for the red and blue groups, even with a soft margin. However if the data are transformed into a three-dimensional space where one of the dimensions was the sum of the squares of the coefficients d^l_k, or the magnitude of $w(x, \omega)$, the component of the signal in the orthogonal space $V_0^\perp$, then clearly there is a more convenient two-dimensional hyperplane that will better separate the data with a soft margin.

This is shown in the following figure where data that tended to appear closer to the origin of projection coefficients, are now generally lower in the new feature space than those that appeared further from the origin. Thereby improving the performance of a decision hyperplane between them.



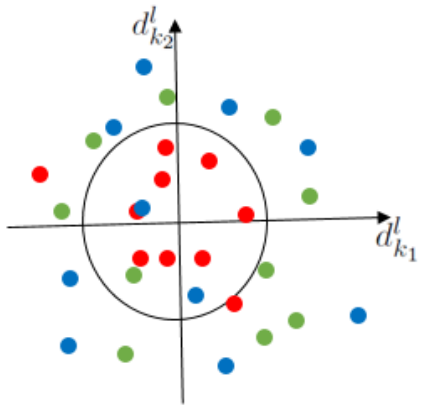
The above illustration is largely for explanatory purposes. It is possible that even with a radial basis function kernel, the decision boundaries may not necessarily appear spherical in the original data space.

The simplest choice for selecting the reference class that will be used to construct the multilevel filter is the class with the largest amount of data. *The paper* implemented this method on data with generally large imbalances in the classes of data. For example, the tumor dataset used had 190 tumor samples and 90 healthy controls. Clearly the tumor class of data has relatively large excess compared to the healthy controls, so it makes sense to use that excess to train the multilevel filter. In fact, it is good to use the entirety of the 100 excess samples that the tumor class has on the healthy controls, that way the remaining data used to train the SVM are balanced. This helps improve the efficiency of the SVM.

In general when it comes to classification methods and comparisons of groups, balanced group designs offer the most efficient use of the data available. The following perspective from a statistics background helps justify this view, unequal allocation of groups in data generally results in suboptimal variance estimates in some form. Hence it is usually the smallest group in a study that limits the power for statistical tests in that study. Therefore the convenience of this method is that a self-determined number of samples from the largest class of data can be used to train the multilevel filter, and this number can be maximized to balance the remaining samples for the SVM classifier.

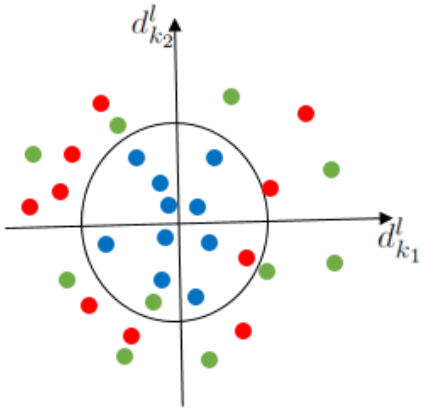
2.7 Multiclass Classification

In order to potentially expand this method to supplement multiclass classification in cases of three or more classes instead of just binary classification, we propose the following framework. Below is a figure very similar to previous, except there are three classes of data presented. Like before, the red class is the reference class that was used to estimate the eigenfunctions and multilevel coefficients. Hence, as we would expect, the red class generally populates the area closer to the origin of the coordinate system composed of the projection coefficients, and the blue and green classes generally populate areas further away from the origin.

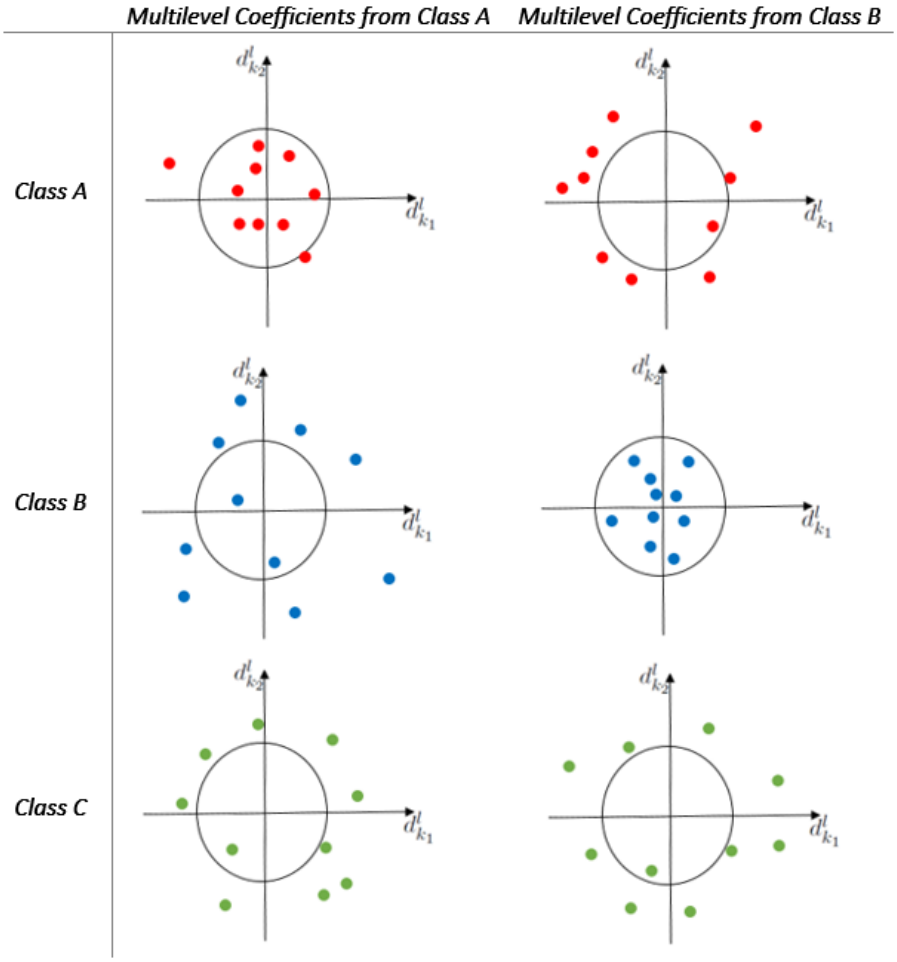


While this is helpful for differentiating blue and green classes from the red class, it is very difficult to distinguish the blue and green classes from each other, as there is no reason for the blue or green classes to be separable in this space, assuming they are distinct processes from the red class. We would expect the green and blue classes to vary similarly across the projection space.

To alleviate this, we propose estimating a second set of eigenfunctions and multilevel coefficients based on the data from a second class such as the blue class in this example. As can be seen in the figure below, now the blue class data generally populate around the origin, and the red and green classes reside largely far away from the origin. We will say this plot is a result of multilevel projections based off of class B, and the previous plot is the result of multilevel projections based off of class A.



Using these two plots, one could create a reasonable decision rule that would be able to classify the three classes distinctly. Below is a table showing the above two plots, but only showing one class at a time.



The idea would be that data from class A would probably be observed near the origin for the multilevel coefficients based on class A, but they would be further from the origin in terms of the multilevel coefficients from class B. Class B would be opposite that of class A. The green class C points would then ideally appear anomalous on both multilevel filters, generally displaying larger coefficients for both the filters based on class A and class B.

One could imagine that this is applicable to a generalizable J number of total classes in a multiclass classification problem, where $J-1$ multilevel filters would need to be created to create a similar structure that allows for each class to be identifiable.

Using this framework, we may feed in the multilevel coefficients from each multilevel filter into a machine learning method that is equipped to handle multiclass classification. Using these coefficients generated from this framework, methods such as SVM may be better equipped and able to improve their classification ability.

To connect to the end of the previous section, the selection of data to be used for training the multilevel filters ought to be driven by the natural imbalance in the data. For example, if classes A, B, and C have 300, 200, and 100 samples respectively, then the limiting set is class C. Hence, the multilevel filters should be trained using classes A and B using 200 and 100 of their samples respectively. This way there are 100 samples in each class remaining for training the SVMs for multiclass classification.

2.8 Simulation Studies

Based on the two SVM frameworks described in the previous section, we implemented multiclass SVM classifiers exploiting the KL expansion multilevel filter described in *the paper*. We also constructed the classifiers without the multilevel filter and compared the difference in accuracies on our self-generated datasets. We generated the data in the following way. Assuming we have 5 classes and 100 distinct features for each sample and 100 samples for each class for training data and 50 samples each class for testing data. Our final training set has is a 500×100 matrix with 500 samples and 100 feature vectors. Our final testing set is a 250×100 matrix. We didn't generate a larger amount of data because of the hardware constraints. The code is executed on the local machines and excess data would cause the runtime to be long. In terms of actual data, we first arbitrarily selected 5 means (-1,2,1,0,3) corresponding to 5 classes denoted $\mu = \mu_k, k \in 1 \ldots 5$. Then for each class, we draw 100 numbers from the normal distribution with $\mu = \mu_k, \sigma^2 = 1$ to be the means of the 100 features of each particular class, denoted $\beta_i, i \in 1 \ldots .100$. Then, for each feature vector X_i for each class, 100 samples for each feature vector x_{ij} is drawn from another normal distribution with $\mu = \beta_i$ and σ^2 being a random real number between 5 to 50, giving us the data for one class. The other classes' data are generated in the same way. This effectively gives us data with 5 classes, each class having its own distribution of 100 features which have a lot of overlap. We assign class 1 to first 1 – 100 samples, class 2 to 101- 200, class 3 to 201 – 300, class 4 for 301-400 and 401 – 500 for class 5. The testing data are generated in the same way as well with sample 1-50 as class 1, 51 – 100 class 2, 101-150 class 3, 151-200 class 4, and 201-250 class 5. The graph below provides a simple illustration of the data generated.

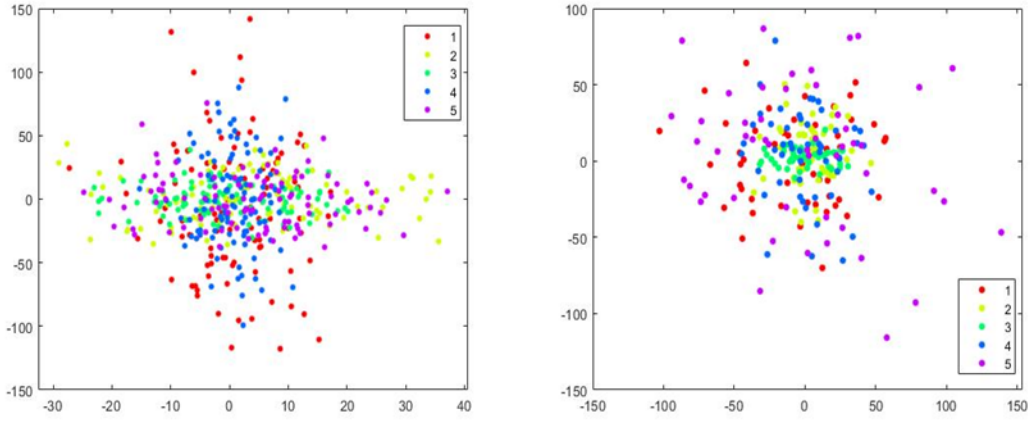
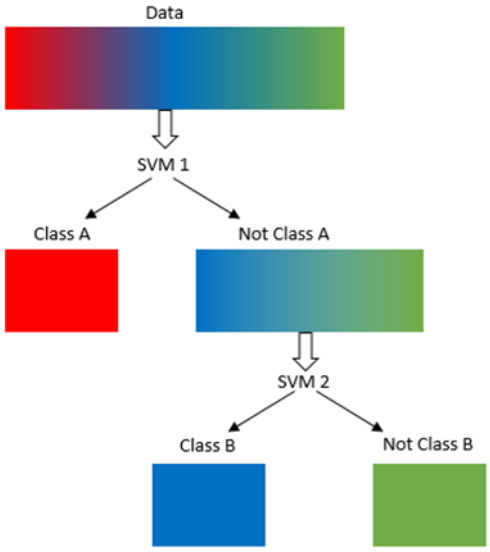


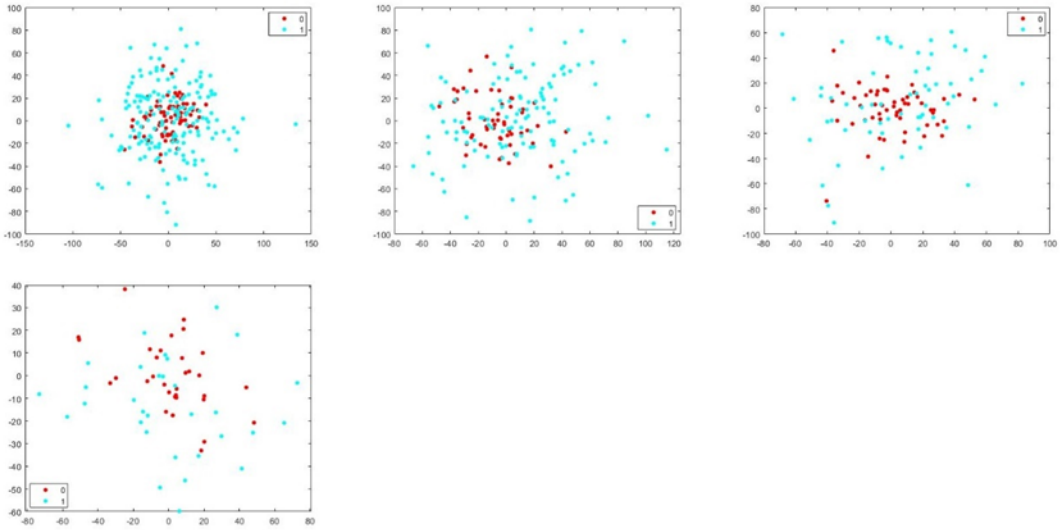
Figure 1: An example of first two feature vectors for generated training and test data

2.9 Greedy Tree Approach

Our first approach to construct our multiclass classifier is by implementing a tree structure classifier. The algorithm works as following: For the training data, we first compute a training accuracy of each class. Assuming we have a balanced data set, we take 80% of the training data of each class to estimate the eigenstructure and construct the multilevel filter for that class. We take the rest 20% of the training data of that as our validation set and select an equal number of samples of the other four classes randomly to obtain a balanced validation set. For our validation set, we first convert them to projection coefficients and then train a binary SVM with RBF kernel. The class we built our multilevel filter on would be class A and other classes would be class B for our binary SVM. The classification goal is to determine whether a sample belongs to class A or if it belongs to some other class B. Due to the small size of our validation data (20 from class A and 20 from class B), we use leave-one-out cross validation and compute the accuracy of one class against all others. We select the best level for our multilevel filter and sequence of classes tested by selecting the level with the highest mean one vs rest accuracy. In terms of ties, we pick the smallest level. We then rank the accuracies of classes from top to bottom to be the sequence of classes tested in the tree classifier. Due to the small number of features in our data, we initialized with max number of orthogonal levels in the multilevel filter as 1 and first 39 largest eigenvalues picked. For the tree classifier, the only difference compared to the previous step is that now for each class we train a one vs all SVM using all validation data. We then classify the testing data by the predetermined sequence of classes in the previous step. If a sample is classified as in the class (class A) at any level. We classify the sample as that class and remove it from the testing data. Thereby creating a tree classifier like structure mentioned in the previous section. We only test the top four classes because all data that's left would automatically be in the last class. An illustration of a tree is placed here as a refresher, so readers don't have to go back and forth.

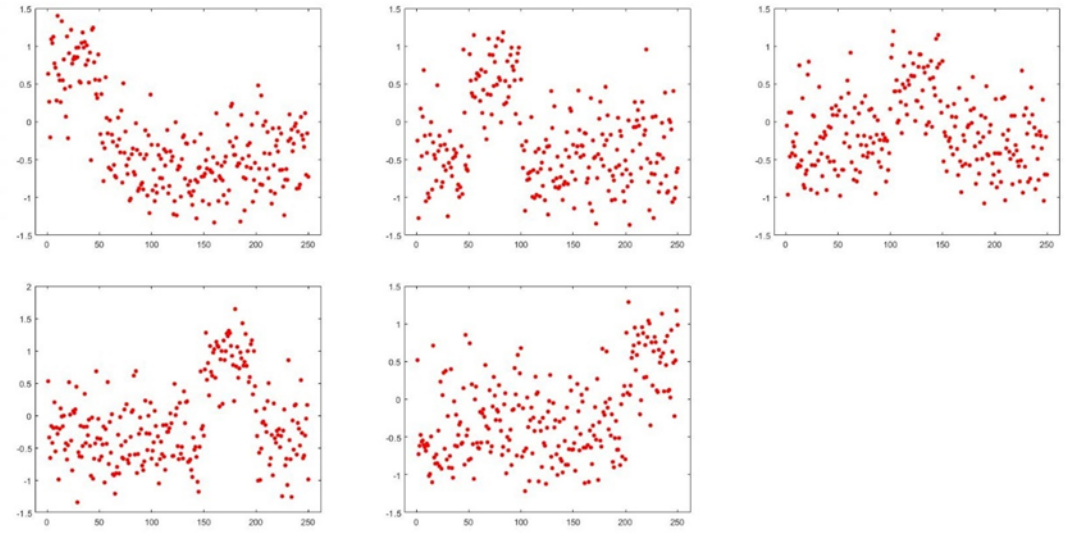


The graph below shows an example classification result at each level of the tree using the simulated data in *Figure 1*, displayed above. We plotted the first two projection coefficients of each testing sample; X-axis is the first projection coefficient and Y-axis the second. The behaviors are as expected. The coefficients for class A, the class which we constructed the multilevel filter on, are clustered in the middle and coefficients of class B, which represents the other classes, are scattered around at the outside.



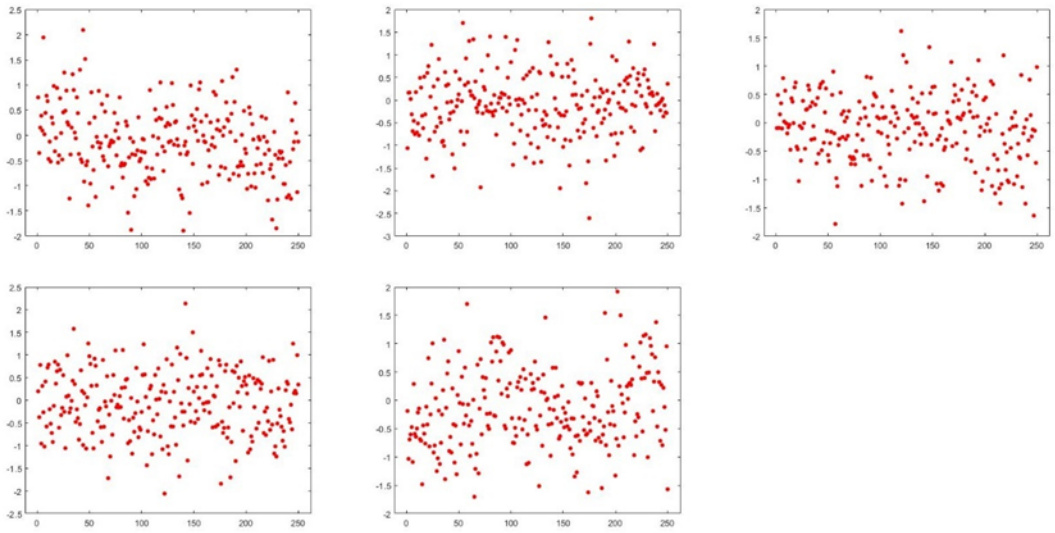
2.10 One-versus-rest SVM with Multilevel Filter

Our second approach is similar to the traditional one-versus-rest SVM classifier approach. We first divide our training set in the similar fashion as in tree-classifier (80% to estimate eigen structure and construct multilevel filter, rest 20% with equal number of samples from other four classes transformed to projection coefficients) and train a one-versus-rest binary SVM with RBF kernel classifier for each class. After we obtain the binary SVM for each class, we take the whole test data and for each class, we transform the testing data to projection coefficients using the corresponding multilevel filter and make predictions using the corresponding SVM trained. There are 5 classes, so we have 5 sets of projection coefficients to be classified using 5 binary SVM classifiers. We therefore obtain 5 sets of likelihoods. We then make the predictions for each sample by comparing the likelihood of classifying as 0, which means that the sample is in the reference class we built the multilevel filter on. The final prediction is the class where the likelihood of being classified as 0 is the highest. For the level of multilevel filter, we use the best level determined in approach 1 to construct the multilevel filter. This approach exploits the property of projection coefficients. If a sample belongs to an arbitrary class A, theoretically the projection coefficients we obtained for this sample using the multilevel filter of class A should be clustering around 0 and we observe a higher likelihood of being classifying into class 0 when classified through the binary SVM of class A vs other classes. And the projection of the same sample on other classes would be far from 0 so it has less likelihood to be classified as "in the class" when classified using SVM of other classes. The graphs below show the likelihood score of being classified as class k, $k = 1, 2, 3, 4, 5$, when classifying the projection coefficients of all testing samples on class k using the corresponding binary SVM of k vs other. The graphs are from left to right with the upper left being SVM class 1 vs other and bottom right being class 5 vs other. Y-axis is the likelihood score, and X-axis is the index of the testing samples from 1-250.



Note that in the data generation process we assigned label 1 to samples 1-50, 2 to 51-100, 3 to 101 – 150, 4 for 151 – 200, and 5 for 201-250. Take the first graph in the above figure as an example. It shows the likelihood score of being classified as class 1 for all testing samples using binary class 1 vs other classes SVM. To obtain the likelihood score plotted in the first graph. We first transform our testing samples into projection coefficients using the multilevel filter constructed from class 1 training data, then use the corresponding SVM of class 1 vs other to obtain the likelihood scores. In this process, we observe that the likelihood scores of

samples 1-50 are particularly high above 0 while samples 51 – 250 have mostly lower than 0 likelihood to be classified as class 1. However, when samples 1-50 are transformed to projection coefficients using multilevel filters constructed using data in other classes k, $k = 2, 3, 4, 5$. The likelihood score for sample 1-50 becomes lower than 0 for most of them using the corresponding k vs other SVM. We observe the similar phenomena for samples from other classes. For example, samples 51-100, which are from class 2, have high likelihood to be classified as class 2 when their projection coefficients onto class 2 are classified using the SVM of class 2 vs other classes in the second graph. The multilevel filter is clearly separating the testing samples after the transformation into projection coefficients. This result also confirms our speculations in the previous multiclass section that a sample from class k, k would be near the origin in terms of the multilevel coefficients based on class k and away from the origin in terms of the multilevel coefficients based on any other classes. We also constructed a one-versus-rest SVM without multilevel filters and results are included below. The one-versus-rest SVM is constructed in the same manner as the second approach with slight difference. Since we no longer estimate the eigenstructure. We use all training data from one class and the same number of samples selected randomly from all other classes to train a one vs all binary SVM with RBF kernel. This process was repeated 5 times corresponding to 5 classes and we have 5 binary SVM classifiers. The likelihood of each sample classified as in the class or not in the class reported and choosing the class where the sample has the highest likelihood to be classified as 0 to be predicted. The test samples' likelihood scores of being classified as class k for all 5 binary k vs other SVM are plotted below.



Note that for SVM without multilevel, we observe no difference between likelihood scores unlike the multiclass classifier using the one-versus-rest with multilevel filter. Showing that the SVM trained with raw data does not classify the data well. Again, take the first graph as an example, which shows the likelihood score of class 1 vs other SVM. While only 1-50 samples belong to class 1, we observe many other samples with higher than 0 score to be classified into class 1. To compare the accuracies on generated training data for our 2 frameworks with/without multilevel filter. We randomly generated 5 sets of data using the methods described above and compared the performance of each classifier.

Trial	Greedy Tree with Multilevel Filter	Multiclass SVM with Multilevel Filter	One-versus-rest SVM without Multilevel Filter	Greedy Tree without Multilevel
1	72.4%	86.4%	40.0%	32.4%
2	70.3%	93.6%	38.4%	33.6%
3	78.0%	92.0%	44.0%	38.4%
4	71.2%	90.8%	36.4%	23.6%
5	70.0%	95.2%	47.6%	41.2%
Mean	72.38%	91.6%	41.28%	33.84%

As we observe, the one-versus-rest SVM with multilevel Filter achieves the best mean accuracy and the frameworks with multilevel filters beat the performance of no multilevel filter by as much as 50%. The data was simulated so that there was a lot of overlap across the class distributions, so it is interesting to see such stark differences in test prediction. The one-versus-rest SVM with multilevel filter also has better mean accuracy compared to greedy tree with multilevel filter. The reason is probably because the greedy tree is biased toward the classes that are classified first. Like we discussed in the previous sections, if a sample is misclassified in the previous levels, there is no chance for it to be reconsidered in the later levels. This problem becomes more severe if the accuracies of the individual SVM are low, which is evident through the abysmal performance of the greedy tree without multilevel coefficients. The one-versus-rest SVM appears to smooth out the problem by comparing the likelihood score of each SVM. Even if one SVM classifies a sample wrong, there is a chance to fix the problem by also considering the outputs from the other SVM. The superiority of the one-versus-rest SVM is also evident through the fact that one-versus-rest SVM without multilevel filter still generally outperforms greedy tree without multilevel filter.

3 Discussion (Theory)

3.1 A Simple Case That Fails the Scheme

Even though the methods based on truncated KL expansions seem robust and free of assumptions, there do exist a variety of cases, which are not complicated or uncommon at all, that can fail the whole scheme. This alarms us of the need for more careful assumption diagnostics to be taken later.

A brief example, visualized in *Figure 2*, goes as follows:

Consider a 2-class distribution in the 3-dimensional feature space. Class c ("c" stands for "class") features:

$$X_c \sim (X_{1_c}, X_{2_c}, X_{3_c}) \text{ where } X_{i_c} \sim \mathcal{N}(\mu_{i_c}, \lambda_{i_c}) \mid i \in \{1,2,3\}, c \in \{A, B\}$$

Specifically, consider the case where we have:

$$\mu_{1_A} = \mu_{2_A} = \mu_{3_A} = \mu_{2_B} = \mu_{3_B} = 0 < \mu_{1_A} << \mu_{1_B}$$

$$\lambda_{1_c} > \lambda_{2_c} > \lambda_{3_c} \textbf{ while}$$

$$\lambda_{1_A} > \lambda_{1_B}, \lambda_{2_A} > \lambda_{2_B}, \lambda_{3_A} > \lambda_{3_B}$$

Setting the cutoff $M = 1$, i.e., ignoring the first axis, class B would look completely "covered" by class A when projected onto any of the remaining axes, which means class A and class B becomes indistinguishable, even if this is a simple case where a LDA or logistic regression can perfectly draw a straight line to seperate the them. The idea that "class A projection coefficients will cluster around 0 because of small eigenvalues" does not hold here. The SVM classification based on these projection coefficients will still be better than a random guess, though, depending on how much the class variance differs on each axis. (SVM will still be able to see that large projection coefficients will more likely be from A, although points around the origin would be heavily confused.)

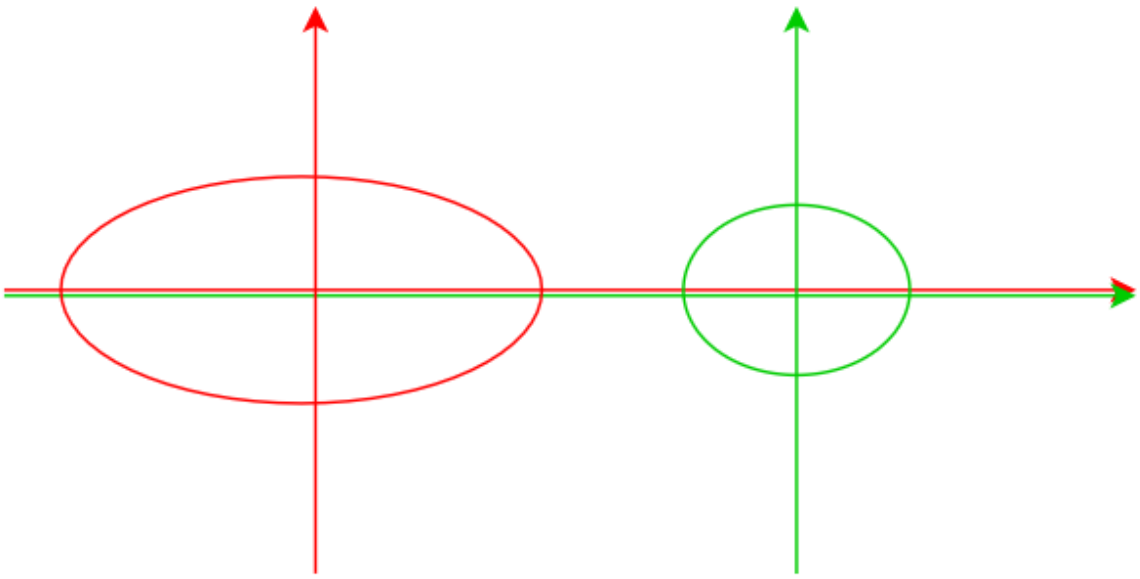


Figure 2: Counterexample diagram demonstrating scheme failure

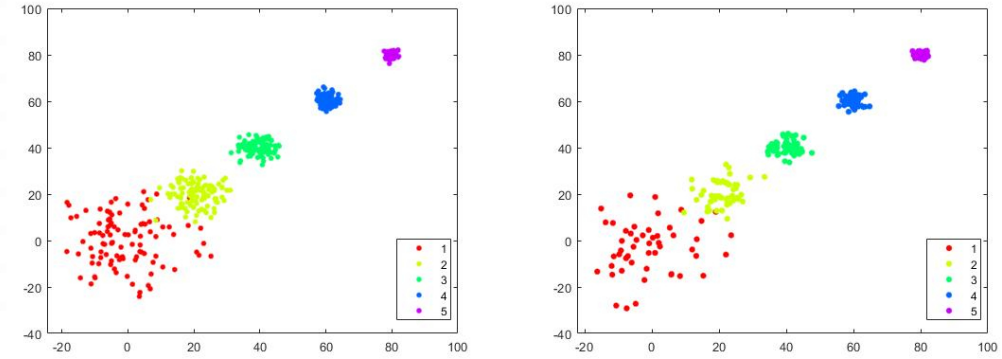
(More precise, strictly controlled simulation studies are still to be finished for better rigor.)

In general, the earlier scheme would fail if the "anomaly" class tends to be clustering even closer to the origin on too many of the extracted axes, because it then brings in confounding variables and "useless" information that "distract" the SVM and pulls down the recognition / learning efficiency.

This explains why, in some early test cases in the simulation studies, SVM would perform even better without the multi-level eigenspace filter, even if the class inputs are generated from ideal, simplistic Gaussian distributions and look perfectly separable.

An example of this is when all features are i.i.d. within the same class. In this case, the whole feature space actually has a low dimensional structure - all class clusters align on a straight diagonal line spanned by the all-1 vector (think of how $\mu_{1_c} = \mu_{2_c} = \ldots$ for all classes c). In this case, all estimated eigenvalues will almost be the same. In fact, in this case, the higher the cutoff M is, and the higher dimensional the feature space is, the larger chance the direction in which these classes separate will be ignored.

Another example is similar. When the data is generated as described in the Methods section, except the range of the variance of features is small, e.g., just 1, which results in a slow decay of eigenvalues in the KL expansion, the cutoff-M scheme actually throws away useful information especially the direction in which the data separate, and deprecate the performance of the kernel SVM. (See appendix for experimental details.)



Trial	Greedy Tree with Multilevel Filter	Multiclass SVM with Multilevel Filter	One-versus-rest SVM without Multilevel Filter	Greedy Tree without Multilevel
1	67.2%	56.8%	100%	100%
2	62.4%	62.4%	100%	100%
3	68.0%	60.0%	100%	100%
4	45.6%	53.2%	100%	100%
5	63.2%	44.8%	100%	100%
Mean	61.28%	55.44%	100%	100%

The mentioned results are from earlier stages of the study. In this example, the mean of the classes are fixed to be (0, 20, 40, 60, 80) and class variance set to be (10,5,3,2,1). Given we have 100 feature vectors, the feature value for class 1 is drawn from the normal distribution with $\mu = 0, \sigma^2 = 10$, class 2 drawn from $\mu = 20, \sigma^2 = 5$, class 3 drawn from $\mu = 40, \sigma^2 = 3$, class 4 from $\mu = 60, \sigma^2 = 2$, and class 5 drawn from $\mu = 80, \sigma^2 = 1$. So we have a dataset that is close to the previously described failing scheme. The later classes are completely "covered" when projected onto any axis while the clusters of data remain perfectly separable from each other. A graph of example training and testing data generated is included below. Again, we did 5 testing runs generating 5 sets of data and ran our 4 approaches to compare the classification accuracies. It is worth noting that the performance of greedy tree and one-versus-rest SVM with multilevel degenerates drastically compared to the previous section while their non-multilevel companions have perfection accuracy for all classes. The result obtained confirmed our claim that this multilevel method can fail under some extreme situations.

3.2 Possible Solutions and Extensions

(Implementations and rigorous simulation studies are yet to be done in this part.)

3.2.1 Basis Selection

First, the ultimate purpose of utilizing eigenspaces corresponding to smaller eigenvalues is to find directions in which the projection coefficients separate between classes. In the end, the idea will be "ignoring axes that do not contribute much to cluster separation", instead of really sticking to the smaller eigenvalues and abandoning the first few basis functions in all cases. The eigenvalues could still help, though, which will be explained in the later section about the curse of dimensionality.

The procedure can be done in the following way:

- Projecting the whole input dataset onto all basis functions from the KL expansion (estimated and pre-stored from training data) of a reference class;
- Looking at statistics (ideally quantile, but this can be too expensive) of the projection coefficients, preserving the basis on which the difference in distribution of the projection coefficients of the reference class and non-reference class(es) exceeds a certain threshold. In the previous counterexample, axis 1 should then be preserved because it brings in a perfect separation.
- This threshold should take into account both the mean and the variance of the projection coefficients of the other classes. The ideology is the same: a large deviation from the origin (i.e., the class center of the reference class), preferably several standard deviations (which is equal to the singular value) away, would more likely be from the other classes. The cutoff can be either soft or hard - a soft one can be a parabolic-like weight on the corresponding basis computed based on the Chebychev bound (along with the confidence level needed).
- Gathering the projection coefficients on all preserved basis as the new features.
- The multilevel decomposition scheme proceeds as usual, with V0 spanned by the selected set of eigenfunctions.

- The final feature can be fed into other classification models, such as SVM and neural networks.

3.2.2 2-Way Comparison and Scoring System: "Is A" minus "not A"

The above process can be done using the KL expansion of every one of the classes. In this process, each class obtains a metric for point similarity to another class based on the projection coefficients, or a weighted sum of them, on the selected basis. When comparing class A against C, the magnitude of projection coefficients from "A vs others" gives a measure of the likelihood of being C, while that from "C vs others" gives a measure of the likelihood of being NOT C. The two parts can be aggregated into a score, and we would like to design the criteria in a way that maximizes the difference between score "is A" and score "not A".

The challenge / drawback of the above design is:

1. This process is computationally expensive, because it involves projection of the whole dataset onto every single basis function (not to mention the further action of iterating through all classes as reference class), which introduces a time complexity proportional to the dimension of the feature space times the number of classes.
2. Being able to estimate the KL expansion from all other classes requires a balanced dataset.

3.2.3 Challenge: "The Curse of Dimensionality"

One issue identified when attempting to choose the cutoff threshold for class separation is the special geometry of high dimensional feature space.

Consider a unit sphere. This characterizes the cluster boundary of a reference class in the coordinate system of its KL basis, if we normalize components on all axes by the standard deviation (which is just the eigenvalue) corresponding to each axis. The points that locate on the "diagonal" positions of the unit sphere, having coordinates $|x_1| = |x_2| = |x_3| = \ldots = a$, have projection coefficients onto any of the axes equal to $a = 1/\sqrt{n}$

where n is the dimension of the feature space. The larger n is, the smaller magnitude a single projection coefficient will be, and the less separation we will be able to see between clusters located at that boundary line and the cluster at the center when projected onto a single axis. From this perspective, it seems it will become harder and harder to find considerably separated clusters of projection coefficients in the high dimensional case.

However, from practical experience, the scenario is a bit uncommon to be seen - a lot of the times, projection coefficients from other classes do seem to be much larger on the axis of smaller eigenvalues (see *Figure 3* in the Appendix), especially for high-dimensional feature spaces. The author of this section believes the likelihood of these cases can be systematically estimated in general - "perfect coverage" as in the earlier counterexample are supposed to be unlikely to happen under high dimensions / with a large degree of freedom.

With all the above considerations in mind, a deeper question to ask will then be:

Why and how could the method be practically robust, in the way that it was implemented in the existing programs by Castrillon-Candas et al, without too much explicit checking and comparison in distributions (of projection coefficient) for classes other than the reference class, while being an distribution-sensitive approach?

3.2.4 Prototype Learning Methods

An unsupervised clustering of training data points within the same class could help us identify separated class centers. A separate KL expansion for each cluster can potentially enable a sharper decay in the eigenvalues due to the fact that cluster points are less spread out in different directions... although this is further adding to the time complexity.

Remark: Note that all the above measures are heavily relying on the assumption that the testing set would have almost exactly the same distribution as the training set. If there exists any shifting feature or a confounding variable that changes the distribution in the testing set, the coordinate system estimated from KL expansion of the training data could become useless, or could even make it worse, as it can confuse the SVM by rotating and stretching the coordinate system in the wrong direction.

3.3 Discussion on Domain Partition and Multiresolution-like Schemes

3.3.1 On Nested Orthonormal Multi-Resolution Decomposition

When it comes to the nature of the output basis functions at different levels, we should note that there does not seem to be a clear relationship between the residual subspaces at different levels yet, except that the ones from the highest level of resolution, i.e., from cells at the leaf nodes, clearly pertain only to signals restricted to the corresponding cell. Therefore, the larger the maximum level we have, presumably the Which levels work the best though shall still be figured out through cross-validation.

Supposedly, the advantage of domain partition with a multi-resolution scheme comes from the following aspects: 1. Localized information 2. Some kind of "denoising" by getting rid of influence from far away points in domain 3. Dimension reduction - only a subset of basis from the residual eigenspace is considered at a time.

3.3.2 On Similar Approaches and Extensions

Related methods and possible generalizations arise by analogy. Potentially, we may also:

- Obtain the cells with some overlap, which makes it more similar to a moving-window cut. This way the cells within the same level no longer form a domain "partition", and the orthogonality between cells is also ruined, but the points at the cell boundaries get to be attended slightly better in the sense that many simplices right next to them will not always be thrown into a different cell then (which leaves no chance for them to get combined into the same cell, at least at a fixed level of resolution, even if they might be strongly correlated as they are so close in domain).
- Assign different weights to different simplices within a cell (e.g., less weights towards the boundary and more towards the cell center).
- Reshuffle the data entries (basically changing their order) so that "far away" entries / simplices get a chance to be grouped together, not just the ones close by (this should be considered based on the nature / geometry of the input data, like whether there is long-term dependency in the input features). For example, for genetic data it might make better sense to focus on local correlations, since genes that are close together on a chromosome tend to be inherited together simply due to their proximity.

In a broader sense, we could see the analogy of the studied method to the windowing methods in Signal Processing and the attention mechanism in Machine Learning (which arises from the Natural Signal Processing but). It is also intriguing how the seemingly unrelated areas share similar wisdoms.

4 Discussion (Application, in Extended Scenarios)

4.1 Dataset: Vowel Data

The attempted implementations and development for ideas discussed in the previous section was started on a multi-class dataset that possesses several properties that are different from the dataset typically involved in applications in [1]. Issues and discoveries are briefly illustrated below.

4.1.1 Dataset Characteristics [3,4]

One major characteristic of this data set is that the data set contains a specified training / testing grouping. The input data contains reflection coefficients extracted from a Linear Predictive Analysis on several window frames of a vowel pronunciation recorded by about 4 different females and 4 males in the training and testing sets, respectively. The data in the testing set came from a different group of individuals than that in the training set, which introduces potential differences in feature distributions across individuals and across gender.

This actually reminds us of one practical challenge:

All discussions in the previous section are based on the assumption that the testing set and the training set have the same distribution within each class. However, this is not necessarily true in reality, especially when it comes to time-series analysis, such as the bike sharing dataset used as examples in lab projects in MA 575 Linear Models at BU.

Another notable fact of this dataset is it involves a relatively low-dimensional feature space, with only 10 dimensional, as opposed to that in the genetic data which has over 16k input features. The low-dimensional feature space has led to the situation that the number of classes is more than the dimension of the feature space, which makes the basis selection approach described earlier necessary.

4.1.2 Neural Network Models

Since the purpose of obtaining projection coefficients from the KL basis is to augment existing Machine Learning models, we tested the vowel dataset on the popular model, neural networks, first. Both shallow ones and slightly deeper ones were attempted, with dropout layers and batch-normalization layers for overfitting prevention. (See appendix for tested neural network structures.)

Main Observation:

The overall test accuracies are around 55%, while the training accuracies are generally around 90%, after a 300-epoch training. Notably, optimal recognition rates for some classes are not achieved in the same epochs as other classes, which indicates that separating the training process and models for different classes might lead to better results. This also motivates the development of a decision-tree structure to augment classification models as described in the Methods section.

4.1.3 Multi-Class SVM Models

Main Observation:

- Including projection coefficients onto KL basis as additional features enables a precise, accurate learning of the input class distribution, which also leads to heavy overfitting and bias toward the training set as the testing set has a different distribution.
- The value of RBF kernel parameters `C` and `gamma` also makes a huge difference, which reminds us of more careful selection of them in the training process, which can generally only be achieved by a grid search.
- Also, a tendency we are seeing is that including KL projection coefficients as class features makes the class much more sensitive to mean (class center) shifts in the testing set, even if it is just a 1-to-1 coordinate transform (stretches and rotations only) without information loss, which is counterintuitive and calls for our deeper understanding of the learning mechanism of RBF SVM.

(See code appendix for experiment details.)

4.2 Other Future Directions

Some possible future directions include better developing our understanding of the theory in the topics of simplices and the kd-tree which was a major source of frustration. It may also be beneficial to conduct more simulation studies where the distributional assumptions of the data are different, and perhaps we should also see what happens when some covariance structure is imposed on the data. *The paper* also does not discuss much in terms of selection for the parameters M and N_0, the number of orthogonal subspaces W_l to be used in the multilevel filter. While M was largely selected based on cross-validation, there did not appear to be much discussion about N_0. It would be interesting to develop a more robust methodology for assigning N_0 in addition to M.

A particularly ambitious direction that we originally planned on taking but ended up not being able to do was a complete implementation of these techniques in R. The challenges in the other parts of this project had us put this idea on the backburner.

5 Authors' Contributions

YL and TY brought the project to the group. All authors contributed to understanding of the two main papers. AT and LZ contributed most of the re-interpretation of the analytical arguments and re-writing/filling in detail for the arguments from *the papers*. All authors brainstormed methods of implementing the technique in multiclass classification setting. YL and TY contributed most of the implementation of the programming implementations. AT and LZ contributed majority of drafting of report. All authors contributed to, read, and approved the final manuscript

6 References

[0] Github Group Repository (2022). Change_Detection_Machine_Learning: Anomaly Detection Enhancement using KL expansions, https://github.com/yulinl2/Change_Detection_Machine_Learning

[1] Julio E. Castrillón-Candás, Dingning Liu and Mark Kon. Stochastic functional analysis with applications to robust machine learning

[2] Julio E. Castrillón-Candás and Mark Kon. Anomaly detection: A functional analysis perspective, Journal of Multivariate Analysis, Volume 189, 2022, 104885, ISSN 0047-259X

[3] D. H. Deterding, 1989, University of Cambridge, "Speaker Normalisation for Automatic Speech Recognition", submitted for PhD.

[4] A. J. Robinson, 1989, Cambridge University Engineering Department, "Dynamic Error Propagation Networks".

[5] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15). JMLR.org, 448–456.

[6] Wang, Limin. (2008). Karhunen-Loeve expansions and their applications..

[7] C. Schwab, R.A. Todor, Karhunen–Loève approximation of random fields by generalized fast multipole methods, J. Comput. Phys. 217 (1) (2006) 100–122, Uncertainty Quantification in Simulation Science.

[8] Cortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks" (PDF). Machine Learning. 20 (3): 273–297. CiteSeerX 10.1.1.15.9362. doi:10.1007/BF00994018. S2CID 206787478.

[9] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, 1998.

[10] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann and Yasemin Altun (2005), Large Margin Methods for Structured and Interdependent Output Variables, JMLR, Vol. 6, pages 1453-1484.

7 Appendix

Attempted Neural Network Structures

```
# A probability model
mod_idx = '11_hidden'
my_model[mod_idx] = tf.keras.models.Sequential([
  # tf.keras.layers.Flatten(input_shape=(10,)),
  tf.keras.layers.Dense(11, activation='relu'),
  tf.keras.layers.Dropout(0.2), # overfitting prevention
  tf.keras.layers.Dense(12, activation='softmax')
])
my_model[mod_idx].compile(optimizer='adam',
                          loss='sparse_categorical_crossentropy',
                          metrics=['accuracy'])
```

```
# A probability model
mod_idx = '22_hidden'
my_model[mod_idx] = tf.keras.models.Sequential([
  # tf.keras.layers.Flatten(input_shape=(10,)),
  tf.keras.layers.Dense(22, activation='relu'),
  tf.keras.layers.Dropout(0.3), # overfitting prevention
  tf.keras.layers.Dense(12, activation='softmax')
])
my_model[mod_idx].compile(optimizer='adam',
                          loss='sparse_categorical_crossentropy',
                          metrics=['accuracy'])
```

```
# A probability model
mod_idx = '88_hidden'
my_model[mod_idx] = tf.keras.models.Sequential([
  # tf.keras.layers.Flatten(input_shape=(10,)),
  tf.keras.layers.Dense(88, activation='relu'),
  tf.keras.layers.Dropout(0.2), # overfitting prevention
  tf.keras.layers.Dense(12, activation='softmax')
])
my_model[mod_idx].compile(optimizer='adam',
                          loss='sparse_categorical_crossentropy',
                          metrics=['accuracy'])
```

```
# A probability model
mod_idx = 'deeper'
my_model[mod_idx] = tf.keras.models.Sequential([
  # tf.keras.layers.Flatten(input_shape=(10,)),
  tf.keras.layers.Dense(32, activation='relu'),
  tf.keras.layers.Dropout(0.2), # overfitting prevention
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dropout(0.2), # overfitting prevention
  tf.keras.layers.Dense(32, activation='relu'),
  tf.keras.layers.Dropout(0.2), # overfitting prevention
  tf.keras.layers.Dense(12, activation='softmax')
])
my_model[mod_idx].compile(optimizer='adam',
                          loss='sparse_categorical_crossentropy',
                          metrics=['accuracy'])
```

```
mod_idx = 'deeper_more_dropped'
my_model[mod_idx] = tf.keras.models.Sequential([
  # tf.keras.layers.Flatten(input_shape=(10,)),
  tf.keras.layers.Dense(32, activation='relu'),
  tf.keras.layers.Dropout(0.2), # overfitting prevention
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dropout(0.3), # overfitting prevention
  tf.keras.layers.Dense(32, activation='relu'),
  tf.keras.layers.Dropout(0.2), # overfitting prevention
  tf.keras.layers.Dense(12, activation='softmax')
])
my_model[mod_idx].compile(optimizer='adam',
                          loss='sparse_categorical_crossentropy',
                          metrics=['accuracy'])
```

```
mod_idx = 'deeper_with_batchnorm'
my_model[mod_idx] = tf.keras.models.Sequential([
  # tf.keras.layers.Flatten(input_shape=(10,)),
  tf.keras.layers.Dense(32, activation='relu'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dense(32, activation='relu'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dense(12, activation='softmax')
])
my_model[mod_idx].compile(optimizer='adam',
                          loss='sparse_categorical_crossentropy',
                          metrics=['accuracy'])
```

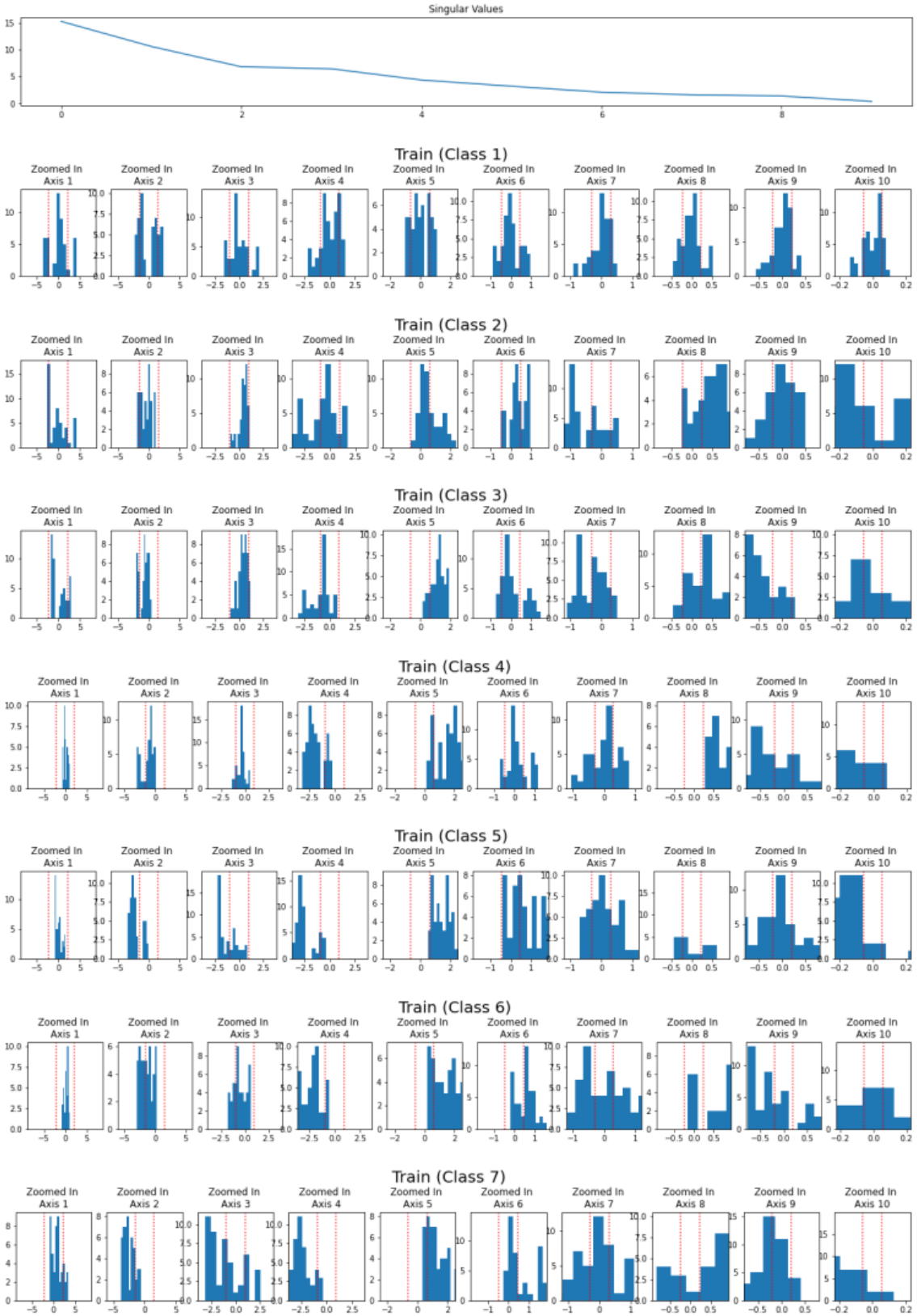


Figure 3: Vowel Dataset Proposed Features Visualization (projection coefficients onto KL basis of Class 1)