Novel IPCA-Based Classifiers and Their Application to Spam Filtering

Alessandro Rozza
Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano
Milan, Italy
rozza@dico.unimi.it

Gabriele Lombardi, Elena Casiraghi
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Milan, Italy
{*lombardi, casiragh*}*@dsi.unimi.it*

***Abstract*—This paper proposes a novel two-class classifier, called `IPCAC`, based on the Isotropic Principal Component Analysis technique; it allows to deal with training data drawn from Mixture of Gaussian distributions, by projecting the data on the Fisher subspace that separates the two classes. The obtained results demonstrate that `IPCAC` is a promising technique; furthermore, to cope with training datasets being dynamically supplied, and to work with non-linearly separable classes, two improvements of this classifier are defined: a model merging algorithm, and a kernel version of `IPCAC`.**

The effectiveness of the proposed methods is shown by their application to the spam classification problem, and by the comparison of the achieved results with those obtained by Support Vector Machines `SVM`, and K-Nearest Neighbors `KNN`.

***Keywords*-Classification; Model-Merging; Kernel Methods; Isotropic PCA**

I. Introduction

In the machine-learning field, the Principal Component Analysis (PCA) is a feature preprocessing step that is often applied both to increase the feature discriminative power, and to decrease the feature space dimensionality.

Nevertheless, PCA is based on the assumption that the input features (points) are drawn from a multivariate Gaussian distribution. To deal with data drawn from Mixtures of Gaussians (MoGs), the Isotropic Principal Component Analysis (IPCA), described in [1], projects the features on the Fisher subspace ($\boldsymbol{FS}$) [2] defined by them. Although in [1] the authors did not report any experimental result, the paper is interesting since it provides new theoretical results that allow to efficiently estimate $\boldsymbol{FS}$.

Exploiting these results, in our work we propose a two-class classification algorithm, called IPCA-based classifier (**IPCAC**, see Section II). As demonstrated by our experiments, the proposed classification algorithm is promising, but it suffers of some limitations: it cannot adaptively manage training sets of high cardinality being dynamically supplied, and it cannot deal with non-linearly separable classes. To overcome these limitations, we have defined two different improvements of IPCAC: the Model-Merging IPCAC (**MM-IPCAC**) and the Kernel IPCAC (**K-IPCAC**).

MM-IPCAC (see Section III) allows to merge IPCAC models, thus dealing with training sets of high cardinality being dynamically supplied, and obtaining for IPCAC advantages similar to those described in [3], [4] for PCA.

K-IPCAC (see Section IV) is a kernel version of IPCAC that allows to overcome the linear separability constraint.

Note that, although the three algorithms are two-class classifiers, they can be generalized to the multi-class case.

To test our methods, we applied them to the TREC email corpus [5] and to the SpamAssassin email corpus [6], with the aim of recognizing "unsolicited bulk" [7] messages (spam) and legitimate messages (ham). The comparison of the achieved results with those obtained by applying well-known classification algorithms, such as Support Vector Machines (SVM, [8]) and K-Nearest Neighbors (KNN, [9]), proves the efficacy of the proposed algorithms.

II. IPCA-based Classifier

Consider a set of clustered points $\boldsymbol{\mathcal{P}} = \{\boldsymbol{\mathcal{P}}_c\}_{c=1}^{C}$ drawn from $\Re^D$, where each cluster $\boldsymbol{\mathcal{P}}_c = \{\boldsymbol{p}_{ci}\}_{i=1}^{N_c}$ contains N_c points (feature vectors). In [2] it is proved that it is possible to find a $(C-1)$-dimensional linear subspace $\boldsymbol{FS_{\mathcal{P}}}$, called the Fisher subspace defined by the given point set $\boldsymbol{\mathcal{P}}$, that minimizes the following discriminant function:

$$J(\boldsymbol{S}) = \frac{\text{intra variance}(\boldsymbol{\mathcal{P}} \text{ proj on } \boldsymbol{S})}{\text{total variance}} = \frac{\mathbb{E}_c\left[\mathbb{E}_i\left[\|proj_{\boldsymbol{S}}(\boldsymbol{p}_{ci}-\boldsymbol{\mu}_c)\|^2\right]\right]}{\mathbb{E}_{ci}\left[\|proj_{\boldsymbol{S}}(\boldsymbol{p}_{ci}-\boldsymbol{\mu})\|^2\right]}$$

where i indexes the points in each cluster, $proj_{\boldsymbol{S}}(\cdot)$ is the linear operator that projects a point on the subspace $\boldsymbol{S}$, $\mathbb{E}.[\cdot]$ is the expectation operator, $\boldsymbol{\mu}$ and $\boldsymbol{\mu}_c$ are respectively the overall mean and the c-cluster mean. Therefore, the Fisher subspace is $\boldsymbol{FS_{\mathcal{P}}} = \text{argmin}_{\boldsymbol{S}}(J(\boldsymbol{S}))$.

In [1] Brubaker and Vempala demonstrate that, given a multivariate probability distribution with mean $\boldsymbol{\mu} = \boldsymbol{0}$, and covariance matrix $\boldsymbol{\Sigma} = \sigma\boldsymbol{I}$ (where $\boldsymbol{I}$ is the identity matrix and σ the standard deviation), and given a set of clustered points $\boldsymbol{\mathcal{P}}$ sampled from it, then the subspace spanned by the $\boldsymbol{\mu}_1$-centered cluster means $\{\boldsymbol{\mu}_c - \boldsymbol{\mu}_1\}_{c=2}^{C}$, approximates $\boldsymbol{FS_{\mathcal{P}}}$.

In the two-class classification problem, $\boldsymbol{FS_{\mathcal{P}}}$ is one-dimensional and it is represented with a unit vector $\boldsymbol{F}$

computed as follows:

$$\boldsymbol{F} = \frac{\boldsymbol{\mu}_A - \boldsymbol{\mu}_B}{\|\boldsymbol{\mu}_A - \boldsymbol{\mu}_B\|} \tag{1}$$

where $c \in \{A, B\}$, and $\boldsymbol{\mu}_A$ and $\boldsymbol{\mu}_B$ ($\boldsymbol{\mu}_{A/B} \in \Re^D$) are the means of the training points belonging to the two classes.

In this case, we classify an unknown test point $\boldsymbol{p}$ by projecting it on $\boldsymbol{F}$, through the dot product ($proj_{\boldsymbol{F}}(\boldsymbol{p}) = \boldsymbol{F}\cdot\boldsymbol{p}$), and then thresholding $proj_{\boldsymbol{F}}(\boldsymbol{p})$.

In practice, the probability distribution related to several classification tasks is not mean-centered, and its random variables are correlated. To solve these problems, we preprocess the data by a linear (whitening) transformation[1]. Considering the set of training points $\hat{\boldsymbol{\mathcal{P}}} = \bigcup_{c=1}^{C} \boldsymbol{\mathcal{P}}_c = \{\boldsymbol{p}_i\}_{i=1}^{N}$, the whitening matrix $\boldsymbol{W}$ is estimated as follows:

1) estimate the expectation $\tilde{\boldsymbol{\mu}} = N^{-1}\sum_i \boldsymbol{p}_i$, and the covariance matrix $\tilde{\boldsymbol{\Sigma}} = N^{-1}\sum_i(\boldsymbol{p}_i - \tilde{\boldsymbol{\mu}})(\boldsymbol{p}_i - \tilde{\boldsymbol{\mu}})^T$;
2) estimate the principal components through the covariance matrix Eigen-decomposition $\boldsymbol{X\Lambda X}^T = \tilde{\boldsymbol{\Sigma}}$;
3) estimate the whitening matrix as $\boldsymbol{W} = \boldsymbol{X\Lambda}^{-\frac{1}{2}}\boldsymbol{X}^T$. Note that $\boldsymbol{\Lambda}^{-\frac{1}{2}}$ can be computed by substituting the $\boldsymbol{\Lambda}$ non-zero diagonal elements λ_i with the values $\lambda_i^{-\frac{1}{2}}$.

The whitened training points, calculated through $\tilde{\boldsymbol{\mu}}$ and $\boldsymbol{W}$, are employed to compute the class means $\boldsymbol{\mu}_A$ and $\boldsymbol{\mu}_B$; $\boldsymbol{F}$ is then computed by means of Equation (1). Therefore, given a new point $\boldsymbol{p}$, it is projected on $\boldsymbol{F}$ as follows:

$$proj_{\boldsymbol{F}}(\boldsymbol{p}) = \boldsymbol{F}\cdot\boldsymbol{W}(\boldsymbol{p} - \tilde{\boldsymbol{\mu}}) = \boldsymbol{w}^T(\boldsymbol{p} - \tilde{\boldsymbol{\mu}}) \tag{2}$$

where $\boldsymbol{w} = \boldsymbol{W}^T\boldsymbol{F}$ is the vector used to simultaneously perform the data whitening and projection.

To classify the new point $\boldsymbol{p}$, a threshold γ must be determined, so that if $\boldsymbol{w}\cdot(\boldsymbol{p} - \tilde{\boldsymbol{\mu}}) > \gamma$ than $\boldsymbol{p}$ is classified as belonging to class A, otherwise it is considered as belonging to class B. To estimate the best threshold value γ, we maximize the number of correctly classified points, that is:

$$\gamma = \left\langle \underset{\{\bar{\gamma}\}\subseteq\{\boldsymbol{w}\cdot(\boldsymbol{p}_i - \tilde{\boldsymbol{\mu}})\}}{\operatorname{argmax}} Score(\bar{\gamma}) \right\rangle \tag{3}$$

where the function $Score(\bar{\gamma})$ computes the number of correctly classified training points when $\bar{\gamma}$ is used as threshold, the argmax operator returns the set of thresholds $\{\bar{\gamma}\}$ leading to the maximum value, and $\langle\cdot\rangle$ is the mean operator.

This classifier is very simple, fast to be trained, and very fast to be applied. Moreover, denoting with D the feature space dimensionality, the space complexity required to store an IPCA-based classifier is at most $2D + 1$: D real values for the estimated expectation[2] $\tilde{\boldsymbol{\mu}}$ (that will be denoted with $\boldsymbol{\mu}$ in the following), D real values for the weight vector $\boldsymbol{w}$, and one value for the threshold γ.

[1]We call "white data" a set of points extracted from a multivariate probability distribution with $\boldsymbol{\mu} = \boldsymbol{0}$, and $\boldsymbol{\Sigma} = \boldsymbol{I}$.

[2]Note that considering the equivalent thresholding function $\boldsymbol{w}\cdot\boldsymbol{p} > \gamma + \boldsymbol{w}\cdot\tilde{\boldsymbol{\mu}} \triangleq \beta$, the space required to store $\tilde{\boldsymbol{\mu}}$ can be avoided.

The main IPCAC drawback is its linearity, in fact it can distinguish only between linearly separable classes.

III. IPCAC Model Merging

One of the benefits of using IPCAC, is the simplicity of combining several classifiers into a single one having three main advantages: it is more accurate than a single IPCAC; it is simply upgradeable with dynamically supplied training data; it is able to cope with large training sets, since each classifier can be trained on a subset of the data.

To perform the IPCAC model merging, the following informations must be stored for each classifier: $\{\boldsymbol{F}, \boldsymbol{W}, \boldsymbol{w}, \gamma, \boldsymbol{\mu}, \boldsymbol{\mu}_A, \boldsymbol{\mu}_B, N_A, N_B\}$, where N_A and N_B are the numbers of training feature vectors used in each class. We call $\{\boldsymbol{\mathcal{M}}_m\}_{m=1}^{M}$ the classification models to be merged, and we call $\{\boldsymbol{F}_m, \boldsymbol{W}_m, \boldsymbol{w}_m, \cdots\}$ their parts. Moreover, we consider the following quantities:

$$N_m = N_{Am} + N_{Bm}, \qquad N = \sum_m N_m$$
$$N_A = \sum_m N_{Am}, \qquad N_B = \sum_m N_{Bm}$$

At first the new (merged) mean vector $\boldsymbol{\mu}$, and the (merged) whitening matrix $\boldsymbol{W}$, must be estimated for the whitening operation. More precisely, to merge the whitening matrices $\boldsymbol{W}_m$, the corresponding covariance matrices must be calculated. To this aim, we employ the Eigen-decomposition $\boldsymbol{W}_m = \boldsymbol{X}_m\hat{\boldsymbol{\Lambda}}_m\boldsymbol{X}_m^T$, and we compute $\boldsymbol{\Sigma}_m = \boldsymbol{X}_m\hat{\boldsymbol{\Lambda}}_m^{-2}\boldsymbol{X}_m^T$. The merged mean vector $\boldsymbol{\mu}$, and the merged covariance matrix $\boldsymbol{\Sigma}$, are then estimated by means of the following generalizations of the results reported in [4]:

$$\begin{aligned}\boldsymbol{\mu} &= N^{-1}\sum_{m=1}^{M}\boldsymbol{\mu}_m N_m\\ \boldsymbol{\Sigma} &= N^{-1}\sum_{m=1}^{M}\boldsymbol{\Sigma}_m N_m\\ &\quad + N^{-2}\sum_{m=1}^{M}\sum_{l=m+1}^{M} N_m N_l(\boldsymbol{\mu}_m - \boldsymbol{\mu}_l)(\boldsymbol{\mu}_m - \boldsymbol{\mu}_l)^T\end{aligned}$$

To compute the merged whitening matrix $\boldsymbol{W}$ starting from $\boldsymbol{\Sigma}$, we use the Eigen-decomposition $\boldsymbol{\Sigma} = \boldsymbol{X\Lambda X}^T$, and we compute $\boldsymbol{W} = \boldsymbol{X\Lambda}^{-\frac{1}{2}}\boldsymbol{X}^T$.

To estimate $\boldsymbol{\mu}_A$ and $\boldsymbol{\mu}_B$ it must be remembered that the means $\boldsymbol{\mu}_{Am}$ and $\boldsymbol{\mu}_{Bm}$ ($m \in \{1, \ldots, M\}$) are determined on the whitened training points; therefore the merge operation requires the inversion of the whitening process. Being the whitening matrices $\boldsymbol{W}_m$ possibly singular, we use their pseudo-inverses denoted by $\boldsymbol{W}_m^{\dagger}$, and we merge the cluster means as follows:

$$\begin{aligned}\boldsymbol{\mu}_A &= N^{-1}\boldsymbol{W}\sum_m \boldsymbol{W}_m^{\dagger}\boldsymbol{\mu}_{Am}N_{Am}\\ \boldsymbol{\mu}_B &= N^{-1}\boldsymbol{W}\sum_m \boldsymbol{W}_m^{\dagger}\boldsymbol{\mu}_{Bm}N_{Bm}\end{aligned}$$

where N_{Am} and N_{Bm} are the numbers of training data points used to estimate $\boldsymbol{\mu}_{Am}$ and $\boldsymbol{\mu}_{Bm}$.

Given the quantities computed above, the new vector $\boldsymbol{F}$, and the weight vector $\boldsymbol{w}$, are computed as before:

$$\boldsymbol{F} = \frac{\boldsymbol{\mu}_A - \boldsymbol{\mu}_B}{\|\boldsymbol{\mu}_A - \boldsymbol{\mu}_B\|}; \qquad \boldsymbol{w} = \boldsymbol{W}^T \boldsymbol{F}$$

The last model part to be merged is the thresholding value γ; to get it, for each classifier m we compute the point $\hat{\boldsymbol{\gamma}}_m$ on $\boldsymbol{FS}_{\boldsymbol{\mathcal{P}}}$ such that its projection on $\boldsymbol{F}_m$ generates exactly the thresholding value γ_m. We get:

$$\begin{aligned} \gamma_m = \boldsymbol{F}_m^T \boldsymbol{W}_m(\hat{\boldsymbol{\gamma}}_m - \boldsymbol{\mu}_m) \quad &\Rightarrow \quad \gamma_m \boldsymbol{F}_m = \boldsymbol{W}_m(\hat{\boldsymbol{\gamma}}_m - \boldsymbol{\mu}_m) \\ &\Rightarrow \quad \hat{\boldsymbol{\gamma}}_m = \gamma_m \boldsymbol{W}_m^\dagger \boldsymbol{F}_m + \boldsymbol{\mu}_m \end{aligned}$$

The merged threshold γ is then computed by projecting the average of the $\hat{\boldsymbol{\gamma}}_m$ on $\boldsymbol{F}$, that is:

$$\begin{aligned} \hat{\boldsymbol{\gamma}} &= N^{-1} \sum_m \hat{\boldsymbol{\gamma}}_m N_m \\ \gamma &= \boldsymbol{F} \cdot \boldsymbol{W}(\hat{\boldsymbol{\gamma}} - \boldsymbol{\mu}) = \boldsymbol{w}^T(\hat{\boldsymbol{\gamma}} - \boldsymbol{\mu}) \end{aligned}$$

Note that, the obtained classification model maintains the same space and time complexity of the original ones.

IV. Kernel IPCAC

To relax the linear separability constraint, imposed by the IPCAC algorithm, it is possible to exploit the kernel trick as in the Kernel Principal Component Analysis (KPCA, [10]), thus obtaining a Kernel Isotropic Principal Component Analysis Classifier (K-IPCAC).

The main idea is that the classes to be separated are non-linearly separable in the original space $\boldsymbol{\mathcal{Q}} = \Re^D$, but it is possible to map the N training points $\boldsymbol{p}_i \in \hat{\boldsymbol{\mathcal{P}}}$ in a higher dimensional space $\boldsymbol{\mathcal{Q}}_\Psi$ where the classes are linearly separable; this is done through a non-invertible map $\boldsymbol{\Psi}(\cdot)$, that is $\boldsymbol{\Psi}(\boldsymbol{p}_i) \in \boldsymbol{\mathcal{Q}}_\Psi$. The generated points are then used to compute the PCA in $\boldsymbol{\mathcal{Q}}_\Psi$, thus obtaining a set of $\bar{N} \leq N$ relevant principal components $\{\boldsymbol{x}_k\}_{k=1}^{\bar{N}} \subset \boldsymbol{\mathcal{Q}}_\Psi$; the subspace spanned by the vectors $\{\boldsymbol{x}_k\}_{k=1}^{\bar{N}}$ is the KPCA subspace where the vectors $\boldsymbol{\Psi}(\boldsymbol{p_i})$ must be finally projected.

In [10] the authors demonstrate that it is possible to compute a weight matrix $\boldsymbol{A} = \{\alpha_{ik}\}_{i,k=1}^{N,\bar{N}}$ that allows to calculate the projection of a point $\boldsymbol{\Psi}(\boldsymbol{p})$ on the principal components $\{\boldsymbol{x}_k\}_{k=1}^{\bar{N}}$. To this aim, in [10] the authors at first assume that the mapped points $\boldsymbol{\Psi}(\boldsymbol{p_i})$ are mean centered in $\boldsymbol{\mathcal{Q}}_\Psi$; in this case the following steps must be performed:

1) compute the design (Gram) matrix $\boldsymbol{K} = \{K_{ik}\}_{i,k=1}^N$ with $K_{ik} = Ker(\boldsymbol{p}_i, \boldsymbol{p}_k)$, where the points $\{\boldsymbol{p}_i\}_{i=1}^N$ are training vectors, and $Ker(\boldsymbol{p}_i, \boldsymbol{p}_k)$ is the kernel function that allows to compute the dot product of $\boldsymbol{\Psi}(\boldsymbol{p}_i)$ and $\boldsymbol{\Psi}(\boldsymbol{p}_k)$;
2) determine the Eigen-decomposition $\boldsymbol{K} = \bar{\boldsymbol{A}}\bar{\boldsymbol{\Lambda}}\bar{\boldsymbol{A}}^T$, and remove eventual zero-variance components obtaining the new decomposition $\boldsymbol{K} = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{\Lambda}}\tilde{\boldsymbol{A}}^T$, where $\bar{N}$ components are retained;
3) compute the weight matrix $\boldsymbol{A} = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{\Lambda}}^{-\frac{1}{2}}$.

Having computed $\boldsymbol{A}$, the generic point $\boldsymbol{\Psi}(\boldsymbol{p})$ can be projected on $\{\boldsymbol{x}_k\}_{k=1}^{\bar{N}}$ as:

$$\begin{aligned} \boldsymbol{x}_k \cdot \boldsymbol{\Psi}(\boldsymbol{p}) &= \sum_{i=1}^N \alpha_{ik} \boldsymbol{\Psi}(\boldsymbol{p}_i) \cdot \boldsymbol{\Psi}(\boldsymbol{p}) \\ &= \sum_{i=1}^N \alpha_{ik} Ker(\boldsymbol{p}_i, \boldsymbol{p}) \end{aligned} \tag{4}$$

When the training points $\boldsymbol{\Psi}(\boldsymbol{p}_i)$ are not mean centered in $\boldsymbol{\mathcal{Q}}_\Psi$, the algorithm described above cannot be directly applied; therefore, as shown in [10], to calculate $\boldsymbol{A}$ the mean centered matrix $\tilde{\boldsymbol{K}}$ must be employed instead of $\boldsymbol{K}$. $\tilde{\boldsymbol{K}}$ is obtained as follows:

$$\tilde{\boldsymbol{K}} = \boldsymbol{K} - \boldsymbol{1}_N \boldsymbol{K} - \boldsymbol{K} \boldsymbol{1}_N + \boldsymbol{1}_N \boldsymbol{K} \boldsymbol{1}_N \tag{5}$$

where $\boldsymbol{1}_N = \{N^{-1}\}_{i,k=1}^N$.

Exploiting these theoretical results, we derived a method to compute the Fisher subspace on training data projected on the KPCA subspace. To describe our method we start by considering a training set mean centered in $\boldsymbol{\mathcal{Q}}_\Psi$, and noting that Equation (4) can be restated in matrix form as:

$$\{\boldsymbol{x}_k \cdot \boldsymbol{\Psi}(\boldsymbol{p})\}_{k=1}^{\bar{N}} = \boldsymbol{A}^T \{Ker(\boldsymbol{p}_i, \boldsymbol{p})\}_{i=1}^N = \boldsymbol{A}^T \boldsymbol{Ker}(\boldsymbol{p})$$

The first step of our method obtains the isotropic components of each training point $\boldsymbol{p_i}$, by using as scaling factor the inverse square root of the diagonal elements λ_k of $\boldsymbol{\Lambda}$ (where $\boldsymbol{\Lambda} = \tilde{\boldsymbol{\Lambda}} N^{-1}$ as shown in [10]), that is:

$$\{\lambda_k^{-\frac{1}{2}} \boldsymbol{x}_k \cdot \boldsymbol{\Psi}(\boldsymbol{p})\}_{k=1}^{\bar{N}} = \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{A}^T \boldsymbol{Ker}(\boldsymbol{p}) \tag{6}$$

Next, we represent the projection of the training points $\hat{\boldsymbol{\mathcal{P}}} \subset \boldsymbol{\mathcal{Q}}$ on the principal components $\{\boldsymbol{x}_k\}_{k=1}^{\bar{N}} \subset \boldsymbol{\mathcal{Q}}_\Psi$ with a matrix $\boldsymbol{P}_\Psi$ obtained as follows:

$$\begin{aligned} \boldsymbol{P}_\Psi &= \{\lambda_k^{-\frac{1}{2}} \boldsymbol{x}_k \cdot \boldsymbol{\Psi}(\boldsymbol{p}_i)\}_{i,k=1}^{N,\bar{N}} \\ &= \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{A}^T \boldsymbol{K} \\ &= (\tilde{\boldsymbol{\Lambda}} N^{-1})^{-\frac{1}{2}} (\tilde{\boldsymbol{A}} \tilde{\boldsymbol{\Lambda}}^{-\frac{1}{2}})^T (\tilde{\boldsymbol{A}} \tilde{\boldsymbol{\Lambda}} \tilde{\boldsymbol{A}}^T) \\ &= N^{\frac{1}{2}} \tilde{\boldsymbol{\Lambda}}^{-\frac{1}{2}} \tilde{\boldsymbol{\Lambda}}^{-\frac{1}{2}} \tilde{\boldsymbol{A}}^T \tilde{\boldsymbol{A}} \tilde{\boldsymbol{\Lambda}} \tilde{\boldsymbol{A}}^T \\ &= N^{\frac{1}{2}} \tilde{\boldsymbol{A}}^T \end{aligned} \tag{7}$$

being $\tilde{\boldsymbol{A}}^T \tilde{\boldsymbol{A}} = \boldsymbol{I}$ for the orthogonality of $\tilde{\boldsymbol{A}}$. Finally, the Fisher subspace is calculated by employing the cluster means of the points represented by the columns of $\boldsymbol{P}_\Psi$.

To relax the hypothesis about mean centering of the training points in $\boldsymbol{\mathcal{Q}}_\Psi$, we consider the projections of the centered points $\{\boldsymbol{\Psi}(\boldsymbol{p}_i) - \boldsymbol{\mu}_\Psi\}_{i=1}^N$, and we exploit the result reported in Equation (5). More precisely, calling $\boldsymbol{\mu}_\Psi =$

$N^{-1}\sum_i \boldsymbol{\Psi}(\boldsymbol{p}_i)$ the mean of the training points mapped in $\boldsymbol{\mathcal{Q}}_\Psi$, we compute the matrix $\boldsymbol{P}_\Psi$ as follows:

$$\begin{aligned}
\boldsymbol{P}_\Psi &= \left\{\lambda_k^{-\frac{1}{2}}\boldsymbol{x}_k \cdot (\boldsymbol{\Psi}(\boldsymbol{p}_i) - \boldsymbol{\mu}_\Psi)\right\}_{i,k=1}^{N,\bar{N}} \qquad (8)\\
&= \left\{\sum_j \lambda_k^{-\frac{1}{2}}\alpha_{ik}(\boldsymbol{\Psi}(\boldsymbol{p}_i) - \boldsymbol{\mu}_\Psi)\cdot(\boldsymbol{\Psi}(\boldsymbol{p}_j) - \boldsymbol{\mu}_\Psi)\right\}_{i,k=1}^{N,\bar{N}}\\
&= \left\{\sum_j \lambda_k^{-\frac{1}{2}}\alpha_{ik}\left(\boldsymbol{\Psi}(\boldsymbol{p}_i)\cdot\boldsymbol{\Psi}(\boldsymbol{p}_j)\right)\right\}_{i,k=1}^{N,\bar{N}}\\
&\quad -\left\{\sum_j \lambda_k^{-\frac{1}{2}}\alpha_{ik}\left(\boldsymbol{\Psi}(\boldsymbol{p}_i)\cdot\boldsymbol{\mu}_\Psi\right)\right\}_{i,k=1}^{N,\bar{N}}\\
&\quad -\left\{\sum_j \lambda_k^{-\frac{1}{2}}\alpha_{ik}\left(\boldsymbol{\mu}_\Psi\cdot\boldsymbol{\Psi}(\boldsymbol{p}_j)\right)\right\}_{i,k=1}^{N,\bar{N}}\\
&\quad +\left\{\sum_j \lambda_k^{-\frac{1}{2}}\alpha_{ik}\left(\boldsymbol{\mu}_\Psi\cdot\boldsymbol{\mu}_\Psi\right)\right\}_{i,k=1}^{N,\bar{N}}\\
&= \boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{A}^T(\boldsymbol{K} - \boldsymbol{1}_N\boldsymbol{K} - \boldsymbol{K}\boldsymbol{1}_N + \boldsymbol{1}_N\boldsymbol{K}\boldsymbol{1}_N)\\
&= \boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{A}^T\tilde{\boldsymbol{K}} = N^{\frac{1}{2}}\tilde{\boldsymbol{\Lambda}}^{-\frac{1}{2}}\tilde{\boldsymbol{\Lambda}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}}\tilde{\boldsymbol{\Lambda}}\tilde{\boldsymbol{A}}^T = N^{\frac{1}{2}}\tilde{\boldsymbol{A}}^T
\end{aligned}$$

Note that $\tilde{\boldsymbol{K}} = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{\Lambda}}\tilde{\boldsymbol{A}}^T \neq \boldsymbol{K}$, and the result reported in Equation (8) is the same as that reported in Equation (7) .

Assuming that the first N_A column vectors $\boldsymbol{P}_{\Psi|1..N_A}$ of $\boldsymbol{P}_\Psi$ belong to class A, and that the remaining column vectors $\boldsymbol{P}_{\Psi|N_A+1..N}$ belong to class B, and noting that these points are whitened through the KPCA algorithm, it is possible to use them for a direct Fisher subspace estimation. To this aim, we must compute the quantities:

$$\begin{aligned}
\boldsymbol{\mu_{A\Psi}} &= \langle\boldsymbol{\Psi}(\boldsymbol{p}_i) - \boldsymbol{\mu}_\Psi\rangle_{i=1}^{N_A}\\
\boldsymbol{\mu_{B\Psi}} &= \langle\boldsymbol{\Psi}(\boldsymbol{p}_i) - \boldsymbol{\mu}_\Psi\rangle_{i=N_A+1}^{N}
\end{aligned}$$

and their difference:

$$\begin{aligned}
\bar{\boldsymbol{F}} &= \langle\boldsymbol{\Psi}(\boldsymbol{p}_i) - \boldsymbol{\mu}_\Psi\rangle_{i=1}^{N_A} - \langle\boldsymbol{\Psi}(\boldsymbol{p}_i) - \boldsymbol{\mu}_\Psi\rangle_{i=N_A+1}^{N}\\
&= \boldsymbol{P}_{\Psi|1..N_A}\Big(\underbrace{N_A^{-1}\cdots}_{N_A \text{ times}}\quad\underbrace{0\cdots}_{N_B \text{ times}}\Big)^T\\
&\quad -\boldsymbol{P}_{\Psi|N_A+1..N}\Big(\underbrace{0\cdots}_{N_A \text{ times}}\quad\underbrace{N_B^{-1}\cdots}_{N_B \text{ times}}\Big)^T\\
&= \boldsymbol{P}_\Psi\Big(\underbrace{N_A^{-1}\cdots}_{N_A \text{ times}}\quad\underbrace{-N_B^{-1}\cdots}_{N_B \text{ times}}\Big)^T = N^{\frac{1}{2}}\tilde{\boldsymbol{A}}^T\boldsymbol{N}_{A|B}^{-1}
\end{aligned}$$

where we have defined $\boldsymbol{N}_{A|B}^{-1} = \Big(\underbrace{N_A^{-1}\cdots}_{N_A \text{ times}}\quad\underbrace{-N_B^{-1}\cdots}_{N_B \text{ times}}\Big)^T$.

The vector $\bar{\boldsymbol{F}}$ must be normalized; its norm is:

$$\begin{aligned}
\|\bar{\boldsymbol{F}}\| &= \left\|N^{\frac{1}{2}}\tilde{\boldsymbol{A}}^T\boldsymbol{N}_{A|B}^{-1}\right\| = N^{\frac{1}{2}}\left\|\boldsymbol{N}_{A|B}^{-1}\right\|\\
&= N^{\frac{1}{2}}\sqrt{N_A\left(N_A^{-2}\right) + N_B\left(N_B^{-2}\right)} = N(N_AN_B)^{-\frac{1}{2}}
\end{aligned}$$

thus, the Fisher subspace can be computed as follows:

$$\boldsymbol{F} = \frac{\bar{\boldsymbol{F}}}{\|\bar{\boldsymbol{F}}\|} = N^{-\frac{1}{2}}(N_AN_B)^{\frac{1}{2}}\tilde{\boldsymbol{A}}^T\boldsymbol{N}_{A|B}^{-1} \qquad (9)$$

In particular, if $N_A = N_B$ we get:

$$\boldsymbol{F} = \frac{\bar{\boldsymbol{F}}}{\|\bar{\boldsymbol{F}}\|} = N^{-\frac{1}{2}}\tilde{\boldsymbol{A}}^T\Big(\underbrace{1\cdots}_{N_A \text{ times}}\quad\underbrace{-1\cdots}_{N_B \text{ times}}\Big)^T$$

Given a testing point $\boldsymbol{p}$, we must compute its projection on $\boldsymbol{F}$; to this aim we use Equation (6) and Equation (9):

$$\begin{aligned}
proj_{\boldsymbol{F}}(\boldsymbol{p}) &= \left(\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{A}^T\boldsymbol{Ker}(\boldsymbol{p})\right)^T(\boldsymbol{F})\\
&= \left(\boldsymbol{Ker}(\boldsymbol{p})^T\tilde{\boldsymbol{A}}\tilde{\boldsymbol{\Lambda}}^{-\frac{1}{2}}\tilde{\boldsymbol{\Lambda}}^{-\frac{1}{2}}N^{\frac{1}{2}}\right)\\
&\quad\left(N^{-\frac{1}{2}}(N_AN_B)^{\frac{1}{2}}\tilde{\boldsymbol{A}}^T\boldsymbol{N}_{A|B}^{-1}\right)\\
&= \boldsymbol{Ker}(\boldsymbol{p})^T\underbrace{\left((N_AN_B)^{\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{\Lambda}}^{-1}\tilde{\boldsymbol{A}}^T\boldsymbol{N}_{A|B}^{-1}\right)}_{\boldsymbol{w}}\\
&= \boldsymbol{Ker}(\boldsymbol{p})^T\boldsymbol{w}
\end{aligned}$$

Note that, since the weight vector $\boldsymbol{w}$ can be precomputed at training time, the classification algorithm is similar to that obtained by Equation (2), that is $\boldsymbol{w}\cdot\boldsymbol{Ker}(\boldsymbol{p}) > \gamma$, where the thresholding value γ is estimated using the same algorithm proposed for IPCAC through Equation (3).

The obtained classifier requires, for each testing point, only N kernel function evaluations more than the IPCAC algorithm, thus remaining very simple and efficient.

V. Results

To test the proposed classification methods we applied them on email classification to recognize spam emails from legitimate emails. In this section we describe the framework implemented for our tests, the experimental setting, and the obtained results.

Our spam filter is a classification system based on email text semantic analysis. The following tasks are performed to achieve the message classification:

Sparse vectors construction: each email is processed to be represented as an high dimensional, and sparse feature vector. More precisely, the words contained in each email are extracted and reduced to the same form by means of standard stemming algorithms [11]; moreover, the obtained set of words is filtered by removing unknown terms (i.e. terms that are not listed in a predefined 'dictionary'[3]). The sparse feature vector is then defined by the frequencies of the remaining words in the processed email.

Sparse to dense projection: to generate an easily manipulable representation, and to extract semantic informations, the sparse feature vectors are projected on

[3]The employed dictionary contains approximately 87000 words and acronyms extracted from different sources.

a lower dimensional space. To this aim, the Term Frequency-Inverse Document Frequency coefficients (`TF-IDF`, [12]), and the Latent Semantic Analysis technique (`LSA`, [13]), are applied to the sparse feature vectors in the training set. These algorithms generate a sparse to dense projection matrix $\boldsymbol{SD}$ that is used to compute all the dense feature vectors processed by the classifiers. We must highlight that sparse to dense projection matrices computed on different training sets allow to compute dense feature vectors of different dimensionality and semantic.

Classification: classification is performed by applying the chosen classifier to the obtained dense feature vectors.

A. Experiments

In order to show the effectiveness of our algorithms, we have compared their results to those obtained by well-known methods described in the literature, more precisely `SVM` and `KNN`[4].

The tests have been performed on two standard email sets, that are: 12000 messages randomly extracted from the TREC 2005 corpus [5] (6000 ham, and 6000 spam), and 3600 messages randomly selected from the SpamAssassin corpus (1800 ham, and 1800 spam) [6]. Each database has been randomly halved, thus obtaining $Dataset1$ and $Dataset2$, and the following two experiments have been executed:

Experiment 1: in this experiment we tested the `IPCAC`, `K-IPCAC`, `SVM`, and `KNN` classifiers by performing 4-fold cross validation on $Dataset1$ and we averaged the evaluation measures.

Experiment 2: this experiment has been performed both to test the robustness of all the classifiers with respect to input vectors generated through different sparse to dense projection matrices, and to compare the `MM-IPCAC` classifier with the other techniques. To this aim the following steps were performed: a sparse to dense projection matrix $\hat{\boldsymbol{SD}}$ was calculated using half of $Dataset1$; the matrix $\hat{\boldsymbol{SD}}$ was employed to process the second part of $Dataset1$; the obtained set of dense vectors was randomly split into four overlapped subsets that were used to train four `IPCAC`, `K-IPCAC`, `SVM`, and `KNN` models. Next, the `MM-IPCAC` technique was applied to merge the four `IPCAC` classifiers. All the trained classifiers were then tested on the feature vectors obtained by processing $Dataset2$ with $\hat{\boldsymbol{SD}}$. The performance of the four `IPCAC`, `K-IPCAC`, `SVM`, and `KNN` classifiers were averaged to obtain their final result.

[4]For each classifier, we determined the best configuration parameters by executing a tuning phase on a smaller data-set, that was automatically generated through random message selection and K-folding.

B. Obtained Results

We executed our tests using the following common indexes as comparison parameters [14]:

$$Accuracy = \frac{100 \cdot (n_{L\to L} + n_{S\to S})}{n_{L\to L} + n_{L\to S} + n_{S\to L} + n_{S\to S}}$$

$$Recall_{spam} = \frac{100 \cdot n_{S\to S}}{n_{S\to L} + n_{S\to S}}$$

$$Precision_{spam} = \frac{100 \cdot n_{S\to S}}{n_{L\to S} + n_{S\to S}}$$

where $n_{L\to L}$ and $n_{S\to S}$ are the correctly classified legitimate and spam messages respectively, while $n_{L\to S}$ and $n_{S\to L}$ are the misclassified legitimate and spam messages respectively.

Experiment$_\#$	*Classifier*	*Accuracy (std)*	*Precision*	*Recall*
Exp$_1$	`KNN`	95.499 (0.759)	95.922	95.028
	`IPCAC`	96.817 (0.279)	96.250	97.430
	`SVM`	97.116 (0.213)	96.234	98.057
	`K-IPCAC`	***97.566*** (0.139)	***96.535***	***98.548***
Exp$_2$	`KNN`	95.449	95.113	95.927
	`IPCAC`	95.05	93.901	96.374
	`SVM`	97.001	95.958	98.133
	`K-IPCAC`	96.917	95.447	98.533
	`MM-IPCAC`	**98.350**	**97.387**	**99.367**

Table I
EXPERIMENTAL RESULTS ON THE EMAILS BELONGING TO TREC CORPUS.

Experiment$_\#$	*Classifier*	*Accuracy (std)*	*Precision*	*Recall*
Exp$_1$	`KNN`	96.278 (0.379)	96.773	95.777
	`IPCAC`	97.444 (1.002)	97.608	97.333
	`SVM`	98.444 (0.602)	98.148	98.778
	`K-IPCAC`	**98.889** (0.181)	**98.569**	**99.222**
Exp$_2$	`KNN`	91.222	96.932	85.111
	`IPCAC`	90.167	93.280	86.667
	`SVM`	93.611	96.444	90.556
	`K-IPCAC`	93.222	94.209	92.111
	`MM-IPCAC`	***98.222***	***98.329***	***98.111***

Table II
EXPERIMENTAL RESULTS ON THE EMAILS BELONGING TO SPAMASSASSIN CORPUS.

Results reported in Tables I and II are commented below:

Experiment 1: on both corpuses `K-IPCAC` outperforms the other classifiers, thus proving that it is the best performing classifier when a single training set is available. Note that the results achieved by `IPCAC` and `SVM` are comparable.

Experiment 2: on both corpuses the results show that our `MM-IPCAC` technique is promising, since it outperforms all the other algorithms.

It is important to underline that all the classifiers, except MM-IPCAC, obtain low accuracy when Experiment 2 is run on the SpamAssassin corpus; this is due to the low cardinality of the subset of $Dataset1$ used to train $\hat{\boldsymbol{SD}}$, producing a loss of informations, relevant for the classification task. Note that, on the TREC corpus, the best accuracy over both the experiments is achieved by the MM-IPCAC algorithm; instead, on the SpamAssassin corpus K-IPCAC obtains the best results, while MM-IPCAC obtains anyway a good performance. Moreover, when Experiment 1 is run on both the corpus, K-IPCAC seams to be the most reliable classifier since it achieves the smallest standard deviation of the accuracy parameter.

These experiments confirm the effectiveness of the proposed algorithms.

VI. Conclusion and Future Work

In this work we defined an IPCA-based classifier (IPCAC), and we applied it to the spam filtering problem. Although promising results were obtained, the proposed classifier cannot manage training datasets being dynamically supplied, and it is based on the assumption that the classes are linearly separable. To overcome these weaknesses we derived two improvements, demonstrating the possibility to merge linear IPCAC models, through the MM-IPCAC technique, and defining a kernel version of IPCAC called K-IPCAC.

The promising results achieved by the proposed classification algorithms, and their comparison with those obtained by well known classification methods, suggest that our techniques can be successfully applied as a basis for more complex learning algorithms.

The IPCAC, MM-IPCAC, and K-IPCAC algorithms are very efficient, and their models are very compact. More precisely, their space and time computational complexity is $O(D)$ (IPCAC, MM-IPCAC) and $O(DN)$ (K-IPCAC), where D and N are the space dimensionality and the number of training samples respectively, as reported in Section II and Section IV.

Future works will aim to generalize our algorithms to the C-class classification problem. This could be the starting point to test the effectiveness of our techniques on other classification issues. Moreover, we plan to develop an efficient K-IPCAC merging algorithm, that would be simultaneously favored by the advantages of both the K-IPCAC and MM-IPCAC methods.

References

[1] S. C. Brubaker and S. Vempala, "Isotropic pca and affine-invariant clustering," *CoRR*, vol. abs/0804.3575, 2008.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience Publication, 2000.

[3] L. Liu, Y. Wang, Q. Wang, and T. Tan, "Fast principal component analysis using eigenspace merging," in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 6, 16 2007-Oct. 19 2007, pp. VI –457–VI –460.

[4] P. Hall, D. Marshall, and R. Martin, "Merging and splitting eigenspace models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, p. 2000, 1998.

[5] G. V. Cormack and T. R. Lynam, "Spam corpus creation for trec," in *CEAS*, 2005.

[6] "Public spamassassin corpus (download page)," 2002-2005, http://spamassassin.apache.org/publiccorpus/.

[7] I. Androutsopoulos, J. Koutsias, K. V. Cb, and C. D. Spyropoulos, "An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages," in *In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM Press, 2000, pp. 160–167.

[8] H. Drucker, S. Member, D. Wu, S. Member, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks*, vol. 10, pp. 1048–1054, 1999.

[9] B. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Soc. Press, 1991.

[10] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, 1998.

[11] J. B. Lovins, "Development of a stemming algorithm." *Massachusetts Inst of Tech Cambridge Electronic Systems Lab*, June 1968.

[12] E. Blanzieri and A. Bryl, "A survey of learning-based techniques of email spam filtering." DIT-06-056, January 2008, university of Trento.

[13] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse Processes*, pp. 259–284, 1998.

[14] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras, and C. D. Spyropoulos, "An evaluation of naive bayesian anti-spam filtering," *CoRR*, 2000.