Feature Extraction through Rotations

Rebeca Salas-Boni and Esteban G. Tabak *

January 24, 2013

Abstract

A procedure is developed for obtaining the lower dimensional representation of high-dimensional observations stemming from different classes that best distinguishes among the classes. The method finds a low dimensional subspace such that the estimated probability of the projected data belonging to each population is as close to the true assignment as possible. This is achieved by starting from a random subspace and applying successive rotations in the direction of descent of the Kullback-Leibler divergence between the true and computed assignments. The method is applied to classification, comparing its performance with benchmark methods on both synthetic data and on the Wisconsin Breast Cancer Diagnostic dataset.

Keywords: Dimensional reduction, Feature Extraction, Gradient Descent, Probabilistic Generative Models, Classification, WBCD.

1 Introduction

Dimensional reduction aims to find an underlying low dimensional structure from high dimensional data. This can be accomplished in two ways, selecting either a subset of the original variables or a map from the high dimensional space of the data onto a lower dimensional manifold. The former one is referred to as feature selection and the latter as feature extraction [4]. In both cases, the reduction sought represents the original data optimally according to a specified criterion. The map of feature extraction can be either linear or non-linear, with linear maps being more widely used. Perhaps the most popular linear dimensional reduction technique, Principal Components Analysis, finds a lower dimensional representation of the data that maximizes the variance of the whole set. Independent Component Analysis seeks projections that are as statistically independent from each other as possible, albeit not necessarily orthogonal as in PCA. There are generalizations of these procedures to the non-linear realm, as well as new methodologies that are non-linear ab-initio.

The techniques above are unsupervised, with no labels taken into account. In classification, by contrast, one has labeled data collected from various populations, from which one

*Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY 10012, USA, salasboni@cims.nyu.edu, tabak@cims.nyu.edu.

aims to learn how to correctly classify new, un-labeled samples. Two main approaches exist for constructing decision boundaries between the classes: geometric methods and probabilistic procedures. The latter estimate the probability of each observation belonging to every possible class taken into account.

Probabilistic methods can be roughly divided in two general approaches: One of them consists in estimating the distributions of the data. Afterwards, the posterior probability of each data point belonging to the different classes is obtained via Bayes' theorem. They suffer from the curse of dimensionality, since the number of observations required to accurately estimate a distribution grows exponentially with the dimension of the data. Hence the importance of reducing the problem's dimensionality. A classic example of a linear dimensional reduction technique is Fisher discriminant analysis [1]. This assumes Gaussian distributions for two classes, and finds the one-dimensional subspace such that, if one projects the multidimensional observations from both classes onto it, the projected observations have their two means as far as possible from each other and are tightly clustered around their corresponding mean. Multilinear discriminant analysis is the generalization of this method to multiple classes: with k classes, the dimension of the observations can be reduced to at most $k-1$. In these methodologies, the data is assumed to stem from Gaussians with different means but the same covariance matrices. Quadratic discriminant analysis incorporates different covariance matrices for each class, which yields a better description of the individual populations but adds more parameters to estimate. In [6], this methodology is extended by modeling observations as Mixtures of Gaussians, allowing for a more accurate characterization of the data.

Non-linear methods to tackle discriminant analysis have also been developed. One such extension works through the introduction of kernels [3], which map the observations non-linearly to a higher dimensional space where the boundary sought is still a hyperplane.

Another popular probabilistic method is the Naive Bayes Classifier (NBC)[2]. Under the assumption that all features are independent from each other, the NBC estimates the probability distribution of each one dimensional feature and computes through Bayes' theorem a posterior probability that the observations belong to each class. Even though the independence of features is a strong assumption, estimating one dimensional probabilities instead of joint ones alleviates the curse of dimensionality and the computational cost.

The second approach for probabilistic methods consists in directly minimizing the difference between the posterior probability of a sample belonging to a population, and the true known assignment. Logistic Regression is an example of such methodolgy. Logistic Regression makes no assumptions on the distribution of the data, rather, this model for classification finds a one-dimensional subspace such that the projected data minimizes an error function. The error function is formed by pairing the known assignments of the data with a posterior probability of each observation belonging to each class, which is seen as the logistic sigmoid of a linear function acting on the observation. This linear function is just the dot product with a vector consisting of weights, which spans the one-dimensional subspace sought. The dependence of the objective function on the vector of weights is non–linear, hence, one must find the subspace through gradient descent [1]. Logistic Regression provides us with a linear decision boundary dividing the classes.

Logistic regression performs a linear combination of the observations, and passes this result through a sigmoid function. Neural networks consist of a network of logistic regressors. Each layer computes a linear combination of the input, and passes it thourgh a sigmoid. When used for classification, the output of the network will be the probability of each observation belonging t each class. Once again, a weights vector is sought, this vector depends non-linearly on the composition of functions, and it is found using gradient descent. Neural networks provide a non-linear decision boundary, the more layers the network has, the more complex the boundary can be. Because of the malleability of the structure of neural networks, feature extraction and classification can be naturally unified [7].
These are just a few of the most commonly used probabilistic methods for classification. However, one can see that dimensionality reduction is somewhat embedded in them. Unfortunately, estimating the distributions of the high–dimensional data will be coslty and imprecise. Nonetheless, having these distributions shed light onto the nature of the data.

The methodology proposed in this article tries to reconcile obtaining a distribution of the data, as well as unloading some of the burden of trying to do so in high dimensions. Our method seeks a low dimensional subspace such that the estimated probability that the projected data belongs to each population is as close to the true assignment as possible. After projecting the observations onto a randomly drawn subspace, one iteratively tilts this subspace so as to minimize the Kullback–Leibler divergence between the estimated posterior probability distributions of the projected data, computed using Bayes' theorem, and the known distribution of the true assignment of the labels, typically composed of zeros and ones. The methodology does not impose restrictions on the target dimension or on the number of classes, other than the need to have enough observations from each class to robustly estimate the corresponding probability density in a subspace of the reduced dimension. It yields a highly flexible algorithm, since one can freely choose the method for estimating the probability distribution of the data. The optimal subspace is found by applying successive small rotations to the data in the direction of descent of the objective function; the sparsity of the matrices involved allows one to efficiently compute and apply the corresponding rotation matrices.

This article describes the algorithm and applies it to synthetic data as well as to the Wisconsin Breast Cancer Diagnostic Classification problem.

2 The general problem

We are provided with N independent n-dimensional observations $x_i \in R^n,\ i = 1, ..., N$, belonging to n_{cl} populations C_j with n_j elements each.

We seek a lower dimensional subspace of specified target dimension $m < n$, such that we can best distinguish among the populations after having projected them onto this subspace. The lower dimensional subspace sought can be described as the subspace spanned by the first m rows of an orthogonal $n \times n$ matrix $\mathcal{O}$.

For all observations x_i, we introduce the following notation:

$$\mathcal{O}(x_i) = \begin{bmatrix} \mathcal{O}_z \\ \mathcal{O}_w \end{bmatrix}(x_i), \qquad \mathcal{O}_z(x_i) =: z_i \quad \mathcal{O}_w(x_i) =: w_i$$

where $\mathcal{O}_z$ and $\mathcal{O}_w$ are $m \times n$ and $(n-m) \times n$ matrices respectively. Our lower dimensional projection of the observations is given by the z_i's.

To measure how well this projection allows us to differentiate among the populations, we compute, using the projected points z_i, estimates for the probability densities $\rho_j(z)$ in each class C_j. Let π_j be the prior probability that an observation belongs to the population C_j (for the given samples, $\pi_j = \frac{n_j}{N}$.) The posterior probability that an observation z belongs to C_j is given by Bayes' formula:

$$P_j(z) = \frac{\pi_j \rho_j(z)}{\sum_l \pi_l \rho_l(z)}.$$

Applied to $z = z_i$, this gives

$$P_j^i := P_j(z_i) = \frac{\pi_j \rho_j(z_i)}{\sum_l \pi_l \rho_l(z_i)}$$

On the other hand, we know independently to which population each observation z_i belongs. We will denote this true distribution by Q, where

$$Q_j^i = \begin{cases} 1 & \text{if } x_i \in C_j \\ 0 & \text{if } x_i \notin C_j \end{cases}$$

We seek the projection that minimizes the distance between the posterior P and the true assignment Q. A natural way of measuring the distance between two probability distributions is given by the Kullback–Leibler divergence

$$D_{KL}(Q||P) := \frac{1}{N} \sum_{i,j} Q_j^i \log\left(\frac{Q_j^i}{P_j^i}\right),$$

which, when Q consists only of ones and zeros, agrees with the cross entropy

$$H(Q,P) = -\frac{1}{N} \sum_{i,j} Q_j^i \log(P_j^i).$$

In view of this, we introduce the objective function

$$M(\mathcal{O}_z) = -D_{KL}(Q||P) = \frac{1}{N} \sum_i \sum_{z_j \in C_i} \log(P_j^i),$$

which depends on the posterior distribution P. Since this in turn depends on the probability density functions ρ_j, which are estimated using the projected data, the function M depends only on the m-dimensional subspace onto which the data are projected, regardless of the

orthonormal basis chosen to represent it. We propose therefore to define the subspace $\mathcal{O}_z$ as the solution to the optimization problem

$$\max_{\mathcal{O}_z} M(\mathcal{O}_z).$$

We illustrate the discussion that follows with some simple low-dimensional examples: samples are generated from three different 2-dimensional Gaussian distributions, and we apply our algorithm to find the 1-dimensional subspace that maximizes M (the details of the algorithm are described in the following section.) We show two different kinds of plots. One corresponds to the function M versus the rotation angle θ that parametrizes the two-dimensional orthogonal matrices $\mathcal{O}$,

$$\mathcal{O}(\theta) = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

The second plot displays the data generated and the line or lines in the direction of the first row of $\mathcal{O}(\theta^*)$ where M achieves its maximum.

A natural question to ask is whether the optimal subspace is unique. Figure 1 displays a situation where uniqueness fails: three identical distributions centered at the vertices of an equilateral triangle have three distinct solutions with the same maximal value of M, as the number of observations grows. Such unlikely situation where two or more subspaces yield the same maximal M does not really pose a problem: all (in this case three) subspaces are equally suitable choices. A more difficult setting for optimization is the presence of local maxima. An instance of this more likely event is represented in figure 2. In view of the possibility of scenarios like this –harder to detect in a higher-dimensional setting–, a strategy needs to be devised to perform a thorough search in the space of candidate subspaces. A procedure to this effect that we implemented is to do various searches, each starting from a different, randomly chosen hyperplane.

A third challenge for the optimization arises from the possibility of M being very pointy close to a local maximum. Intuitively, this reflects a scenario where at least one of the populations has its support concentrated in a neighborhood of a lower dimensional subspace. In the Gaussian case, this corresponds to a covariance matrix with one or more very small eigenvalues. Because of this degeneracy, tilting away slightly from the optimal subspace could translate into a big change in M. We illustrate this situation in figure 3.

In the following two sections, we develop a methodology for maximizing M.

3 Two to one-dimensional case

The simplest scenario for the setting above has two-dimensional observations (x, y) that one seeks to project onto a line:

$$\mathcal{O}(\theta) \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} z \\ w \end{pmatrix},$$

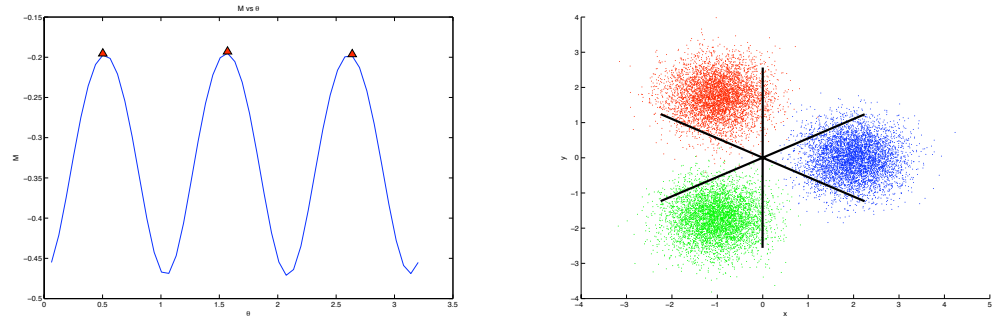

Figure 1: A very symmetric case, where all three Gaussian distributions have the same covariance matrix and their means are equidistant along a circle. A larger (and similar) number of observations drawn form each one of the populations increases the chances of M achieving its maximum at the three different places depicted.

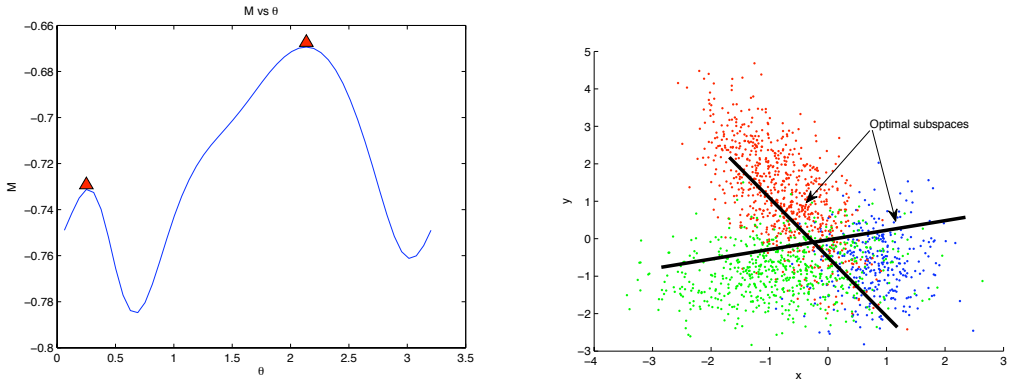

Figure 2: An example where the function M has two local maxima.

so the one-dimensional reduced observation z is given by

$$z = x\cos(\theta) + y\sin(\theta).$$

We will decompose the orthogonal matrix $\mathcal{O}$ into the product of small rotations

$$\mathcal{O} = \cdots \mathcal{O}_{k+1}\mathcal{O}_{k+1}\cdots\mathcal{O}_1\mathcal{O}_0$$

The algorithm that we propose works by ascent, updating at the k-th step z and w through rotations by small angles θ_k:

$$\begin{pmatrix} z \\ w \end{pmatrix}_{k+1} = \mathcal{O}(\theta_k)\begin{pmatrix} z \\ w \end{pmatrix}_k = \begin{pmatrix} \cos(\theta_k) & \sin(\theta_k) \\ -\sin(\theta_k) & \cos(\theta_k) \end{pmatrix}\begin{pmatrix} z \\ w \end{pmatrix}_k.$$

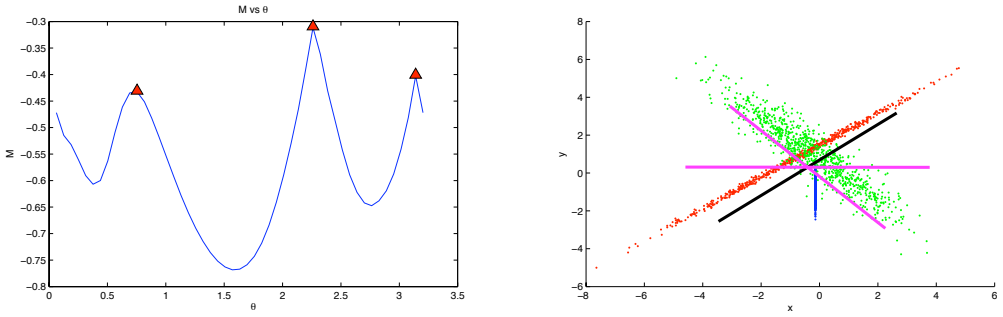

Figure 3: The subspaces corresponding to pointy maxima in M are colored in magenta. Both the red and the blue population were generated with a covariance matrix with one of its singular values close to zero. The magenta subspaces point in a direction close to that of the singular vector corresponding to the very small singular value of one of either the red or the blue populations.

For brevity, we shall omit the subscript k and $k+1$ from here on; the context will make clear which we are referring to. We will use subscripts instead to indicate observations and superscripts for the classes. The small rotation $\mathcal{O}_k$ at each step of the algorithm will follow the direction of ascent of the function we wish to maximize,

$$M = \frac{1}{N} \sum_j \sum_{z_i \in C_j} \log(P_i^j) = \frac{1}{N} \sum_j \sum_{z_i \in C_j} \log \left(\frac{\pi_j \rho_j(z_i)}{\sum_l \pi_l \rho_l(z_i)} \right)$$

We explain how to find the angle θ_k. First, we select a family of probability density functions ρ_j to model the population j of the form $\rho_j(x) = \rho(z, \alpha_j)$, where the parameters α_j are given by the maximum likelihood estimator,

$$\alpha_j = \operatorname{argmax} \sum_{z_i \in C_j} \log \rho_j(z_i, \alpha_j)$$

Differentiating M with respect to θ yields

$$\frac{\partial M}{\partial \theta} = \frac{1}{N} \sum_j \sum_{z_i \in C_j} \left[\frac{P_z^j(z_i)}{P^j(z_i)} \frac{\partial z_i}{\partial \theta} + \frac{P_\alpha^j(z_i)}{P^j(z_i)} \frac{\partial \alpha_j}{\partial \theta} \right]$$

where the subscriptes z and α denote differentiation with respect to z and α.

We examine the components of this sum. The probability P depends on z and α only through the probability density $\rho(z, \alpha)$, which we assume to have an explicit form that one can differentiate with respect to z and α. Hence, we only need to compute the derivatives of the observations and the parameters α with respect to the rotation angle θ. For the observations, this is straightforward:

$$\frac{\partial z_i}{\partial \theta} = -x_i \sin(\theta) + y_i \cos(\theta)$$

which, evaluated at $\theta = 0$, yields

$$\left.\frac{\partial z_i}{\partial \theta}\right|_{\theta=0} = y_i$$

Notice how, instead of computing a new observation z, we can just extract the value y. This alleviates the computations when proceeding to higher dimensions.
In order to find $\frac{\partial \alpha}{\partial \theta}$, we recall that α is given by

$$\alpha_j = \text{argmax} \sum_{z_i \in C_j} \log \rho_j(z_i, \alpha_j),$$

so the optimal α satisfies

$$\sum_{z_i \in C_j} \frac{\rho_\alpha(z_i)}{\rho(z_i)} = 0,$$

where the subscript α indicates differentiation with respect to α and we have omitted the subscript j for clarity. Denoting $F(z_i) := \frac{\rho_\alpha(z_i)}{\rho(z_i)}$ and differentiating the identity above with respect to θ yields

$$\sum_{z_i \in C_j} \left[F_z(z_i) \frac{\partial z_i}{\partial \theta} + F_\alpha(z_i) \frac{\partial \alpha}{\partial \theta} \right] = 0,$$

from which it follows that

$$\frac{\partial \alpha}{\partial \theta} = -\frac{\sum_{z_i \in C_j} F_z(z_i) \frac{\partial z_i}{\partial \theta}}{\sum_{z_i \in C_j} F_\alpha(z_i)},$$

an equation that allows us to express $\frac{\partial \alpha}{\partial \theta}$ in terms of functions known explicitly.

To conclude, since the small rotations $\mathcal{O}_k$ are close to the identity, we obtain the new rotation angle θ_k from

$$\theta_k = \eta \left.\frac{\partial M}{\partial \theta}\right|_{\theta=0}$$

where η is the learning rate. Then we form the rotation matrix and update the projections z_i of the observations x_i.

3.1 An example: Gaussian distributions

To fix ideas, we develop the methodology for Gaussians estimators for the densities ρ_j. From the updated $z_i = z_i \cos(\theta) + w_i \sin(\theta)$, we choose the maximum likelihood Gaussian univariate density estimation

$$\rho_j(z) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2}\left(\frac{z-\mu_j}{\sigma_j}\right)^2},$$

where

$$\mu_j = \frac{1}{n_j} \sum_{i=1}^{n_j} z_i$$

and

$$\sigma_j^2 = \frac{1}{n_j}\sum_{i=1}^{n_j}(z_i - \mu_j)^2\,.$$

the sample mean and variance of the j-th class.

Since

$$\left.\frac{dz_i}{d\theta}\right|_{\theta=0} = w_i,$$

$$\left.\frac{d\mu_j}{d\theta}\right|_{\theta=0} = \frac{1}{n_j}\sum_{i=1}^{n_j} w_i^j =: \mu_j^w$$

and

$$\left.\frac{d\sigma_j}{d\theta}\right|_{\theta=0} = \frac{1}{n_j\sigma_j}\sum_{i=1}^{n_j}\left(z_i^j - \mu_j\right)\left(w_i^j - \mu_j^w\right),$$

it follows that

$$\frac{d\rho_j(z_i)}{d\theta} = \left[-\frac{1}{{\sigma_j}^2}(z_i-\mu_j)\left(w_i-\mu_j^w\right) + \frac{d\sigma_j}{d\theta}\left(-\frac{1}{\sigma_j} + \frac{(z_i-\mu_j)^2}{{\sigma_j}^3}\right)\right]\rho_j(z_i)\,.$$

With this, one can compute the derivative of M with respect to θ:

$$\begin{aligned}
\frac{dM}{d\theta} &= \frac{1}{N}\sum_{j=1}^{n_{cl}}\sum_{z_i\in C_j}\frac{d}{d\theta}\log\left(P_j(z_i)\right)\\
&= \frac{1}{N}\sum_{j=1}^{n_{cl}}\sum_{z_i\in C_j}\left[\frac{d}{d\theta}\log\left(\pi_j\rho_j(z_i)\right) - \frac{d}{d\theta}\log\left(\sum_l \pi_l\rho_l(z_i)\right)\right]\\
&= \frac{1}{N}\sum_{j=1}^{n_{cl}}\left[\sum_{z_i\in C_j}\left(\frac{\frac{d\rho_j}{d\theta}(z_i)}{\rho_j(z_i)} - \left(\frac{\sum_l \pi_l\frac{d\rho_l}{d\theta}(z_i)}{\sum_l \pi_l\rho_l(z_i)}\right)\right)\right]
\end{aligned}$$

and, in the simplest ascent procedure, take

$$\theta = \eta(\epsilon)\frac{dM}{d\theta}, \qquad \eta(\epsilon) = \frac{\epsilon}{\sqrt{\epsilon^2\left|\frac{dM}{d\theta}\right| + \left|\frac{dM}{d\theta}\right|^2}}$$

where the formula for the learning rate $\eta(\epsilon)$ yields steps of size $\epsilon \ll 1$ far from the optimal θ_*, and of size $\sqrt{\left|\frac{dM}{d\theta}\right|}$ near θ_*.

4 Multidimensional case

A more general setting has the original space of features to be n-dimensional, and the sought subspace Z of dimension $m < n$. Still restricting ourselves to linear orthogonal projections, we have

$$\begin{pmatrix} z \\ w \end{pmatrix}_{k+1} = \mathcal{O} \begin{pmatrix} z \\ w \end{pmatrix}_k = \begin{bmatrix} \mathcal{O}_z \\ \mathcal{O}_w \end{bmatrix} \begin{pmatrix} z \\ w \end{pmatrix}_k,$$

where z and w are m and $(n-m)$-dimensional vectors respectively, and

$$\mathcal{O} = \begin{bmatrix} \mathcal{O}_z \\ \mathcal{O}_w \end{bmatrix}$$

is an orthogonal matrix, with $\mathcal{O}_z \in R^{m \times n}$ and $\mathcal{O}_w \in R^{(n-m) \times n}$.

For concreteness, we choose ab-initio the probability densities ρ to be multivariate Gaussians,

$$\rho(z) = \frac{1}{(2\pi)^{\frac{m}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(z-\mu)^\top \Sigma^{-1} (z-\mu)},$$

where the parameters μ and Σ are the maximum likelihood estimators

$$\mu = \frac{1}{n} \sum_i z_i \qquad \Sigma = \frac{1}{n} \sum_i (z_i - \mu)(z_i - \mu)^\top$$

For notational clarity, we eliminate the subscript for the observation and indicate averaging over a population by a bar. Hence the previous equations become:

$$\mu = \overline{z} \qquad \Sigma = \overline{(z-\mu)(z-\mu)^\top}$$

A general orthogonal matrix $\mathcal{O}$ depends on $n(n-1)/2$ parameters: the independent entries of the skew-symmetric matrix A of its Lie-algebra:

$$\mathcal{O} = e^A, \quad \text{with} \quad A^i_j = -A^i_j.$$

When A is small, each entry A^i_j defines a two-dimensional differential rotation in the plane ij. We are not interested in rotations within the Z or W subspaces, since these would only act as re-parameterizations of the subspaces. Then we need only consider the $m(n-m)$ entries where $i \leq m$ and $j > m$ and their skew-symmetric counterparts. Hence, our matrix A has the form

$$A = A(S) = \left(\begin{array}{c|c} 0_{m \times m} & S \\ \hline -S^\top & 0_{(n-m) \times (n-m)} \end{array} \right)$$

where S is an $m \times (n-m)$ matrix.

4.1 Gradient ascent

We compute the gradient of M,

$$G^i_j = \frac{\partial M}{\partial S^i_j},$$

and write, by ascent,

$$\mathcal{O} = e^{A(S)}, \quad S^i_j = \epsilon G^i_j,$$

where ϵ is the learning rate.

Since the entry S^i_j corresponds to a rotation in the plane ij by the angle $\theta = S^j_i$, which we will refer to as $\mathcal{O}_{ij}$, we compute the derivative as follows:

$$\mathcal{O}_{ij}(x) = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos(\theta) & \cdots & \sin(\theta) & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\sin(\theta) & \cdots & \cos(\theta) & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} z \\ w \end{pmatrix} = \begin{pmatrix} \vdots \\ z_i \cos(\theta) + w_j \sin(\theta) \\ \vdots \\ -z_i \sin(\theta) + w_j \cos(\theta) \\ \vdots \end{pmatrix}$$

Hence, computing the gradient G is entirely similar to the derivative M_θ of the two-dimensional case: we will differentiate the function M with respect to θ and set $\theta = 0$. Denoting by z_k the k-th entry of z, we have that

$$\left.\frac{\partial z_k}{\partial \theta}\right|_{\theta=0} = \delta^k_i w_j,$$

where w_j is the j-th entry of w. Then

$$\left.\frac{\partial \mu_k}{\partial \theta}\right|_{\theta=0} = \delta^k_i \overline{w_j},$$

so μ_θ is a vector whose only non-zero entry is located in the i-th place, where we find the average over all the j-th entries of the w's. Likewise for Σ:

$$\begin{aligned} \left.\frac{\partial \Sigma}{\partial \theta}\right|_{\theta=0} &= \overline{(z_\theta - \mu_\theta)(z-\mu)^\top + (z-\mu)(z_\theta - \mu_\theta)^\top} \\ &= \overline{\left[\begin{pmatrix} 0 & \cdots & 0 \\ - & (w_j - \overline{w_j})(z-\mu) & - \\ 0 & \cdots & 0 \end{pmatrix} + \begin{pmatrix} 0 & | & 0 \\ \vdots & (w_j - \overline{w_j})(z-\mu) & \vdots \\ 0 & | & 0 \end{pmatrix} \right]} =: \Sigma_\theta \end{aligned}$$

is a sparse matrix with non-zero entries only along its i-th row and i-th column. Now we can compute

$$\begin{aligned} \left.\frac{\partial \rho}{\partial \theta}\right|_{\theta=0}(z) = \frac{\rho(z)}{2} [&(z-\mu)^\top (\Sigma^{-1}\Sigma_\theta \Sigma^{-1})(z-\mu) \\ &- (z-\mu)^\top \Sigma^{-1}(z_\theta - \mu_\theta) - (z_\theta - \mu_\theta)^\top \Sigma^{-1}(z-\mu) \\ &- \operatorname{tr}(\Sigma^{-1}\Sigma_\theta)] \end{aligned}$$

and finally

$$\begin{aligned}\frac{\partial M}{\partial \theta} &= \frac{1}{N}\sum_{j=1}^{n_{cl}}\sum_{z_i\in C_j}\frac{\partial}{\partial\theta}\log\left(P_j(z_i)\right)\\ &= \frac{1}{N}\sum_{j=1}^{n_{cl}}\sum_{z_i\in C_j}\left[\frac{\partial}{\partial\theta}\log\left(\pi_j\rho_j(z_i)\right)-\frac{\partial}{\partial\theta}\log\left(\sum_l \pi_l\rho_l(z_i)\right)\right]\\ &= \frac{1}{N}\sum_{j=1}^{n_{cl}}\left[\sum_{z_i\in C_j}\left(\frac{\frac{\partial\rho_j}{\partial\theta}(z_i)}{\rho_j(z_i)}-\left(\frac{\sum_l \pi_l\frac{\partial\rho_l}{\partial\theta}(z_i)}{\sum_l \pi_l\rho_l(z_i)}\right)\right)\right].\end{aligned}$$

Replacing θ by S^i_j, we have all the entries of the gradient of M,

$$G^i_j = \frac{\partial M}{\partial S^j_i}$$

The appendix describes how to use the sparsity of the matrices involved to compute $\frac{\partial\rho(z)}{\partial\theta}$ efficiently.

It only remains to compute $\mathcal{O} = e^{A(S)}$, where $S^i_j = \epsilon G^i_j$. It turns out that, given the special structure of A, the matrix $\mathcal{O}$ can also be computed very efficiently. Let $S = U\Sigma V'$ be the SVD decomposition of S. Then, we compute e^A via the following formula:

$$\mathcal{O} = e^A = I + A + \frac{A^2}{2!} + \cdots = \left(\begin{array}{c|c}\mathcal{O}_{x,x} & \mathcal{O}_{x,y}\\ \hline \mathcal{O}_{y,x} & \mathcal{O}_{y,y}\end{array}\right) =$$

$$\begin{pmatrix} 0_m & 0\\ 0 & I_{n-m}\end{pmatrix} + \sum_{j=1}^{m}\begin{pmatrix} u_j & 0\\ 0 & v_j\end{pmatrix}\begin{pmatrix}\cos(\sigma_j) & \sin(\sigma_j)\\ -\sin(\sigma_j) & (\cos(\sigma_j)-1)\end{pmatrix}\begin{pmatrix} u'_j & 0\\ 0 & v'_j\end{pmatrix}, \qquad (1)$$

which only uses the first m columns of V. With this representation, the matrix $\mathcal{O}$ is inexpensive to compute and to apply to the observations.

5 Numerical examples

In this section, we illustrate the performance of the algorithm presented in this paper via several synthetic examples and one real life application.

5.1 Visualization

First, we use a few plots to illustrate graphically the output of the algorithm. We generate synthetic data, drawing between 200 and 400 points from five multivariate distributions in dimensions $n = 20$ and $n = 10$. We then apply our algorithm, seeking an optimal two-dimensional subspace. For reference, we show the projection of the data onto a random

two dimensional space; a second plot shows the data projected onto the plane found by the algorithm.

In Figure 4, we choose five 20-dimensional Gaussians with means uniformly distributed on the sphere of radius 3 and covariance matrices with singular values in intervals ranging from $[0, 1]$ to $[0, 5]$. This yields five distributions with approximately equidistant means and varying shapes and spreads. We show a picture of these data points projected onto a random subspace in the first subfigure, and onto the subspace found by the algorithm in the second, where the five distributions are much more clearly separated.

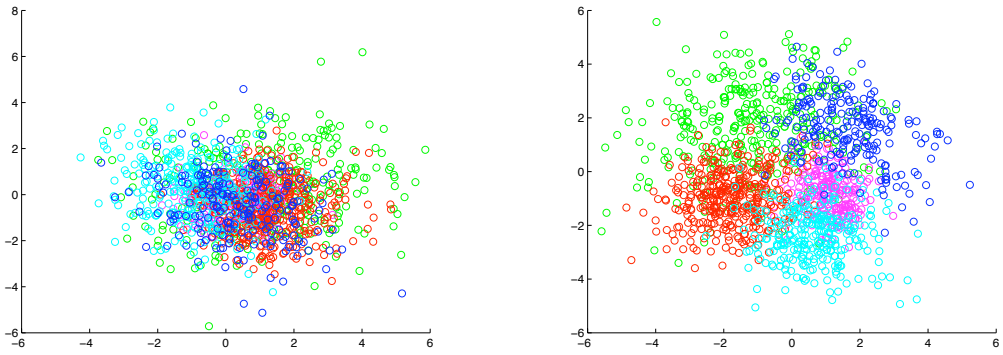

Figure 4: Samples drawn from five 20-dimensional Gaussians, with means on the sphere with radius 3 and singular value ranges varying $[0, 1]$ to $[0, 5]$. On the left, the dataset projected onto a random two-dimensional subspace; on the right, the dataset projected onto the subspace found by the algorithm. Even though some populations are very spread out in some directions, the distance between the means between different populations is large enough to yield good results in few iterations.

In Figure 5, the points are drawn from five 10-dimensional mixtures of two Gaussians. Each mixture has means drawn uniformly from the sphere with radius 3, covariance matrices whose singular values are drawn uniformly from $[0, 1]$ and $[0, .7]$, and mixing coefficients π_1 and π_2 between $1/3$ and $2/3$. The mixed nature of each distribution, clearly noticeable on a random plane, is much less obvious on the optimal one: a byproduct of maximally differentiating the five distributions is to make each relatively compact.

These examples illustrate how the subspace found by the algorithm aids in the visualization of the data. A natural application, developed in the following subsection, is to use this subspace to build a classifier, whose performance then measures the quality and usefulness of the subspace found.

5.2 Numerical example

In order to measure the performance of our method by evaluating its performance as a classifier, we draw samples from two ten-dimensional Multivariate Gaussian distributions, 200 observations from the first and 100 from the second. Both distributions have mean zero.

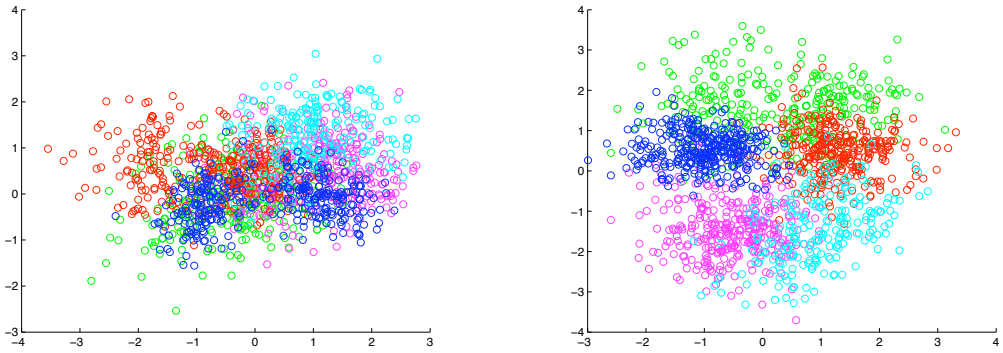

Figure 5: Between 200 and 400 points, drawn from five 10-dimensional Mixtures of two Gaussians, each with means drawn uniformly from the sphere with radius 3, singular values of their covariance matrices drawn uniformly from $[0, 1]$ and $[0, .7]$ and mixing coefficients π_1 and π_2 between $1/3$ and $2/3$. The dataset projected onto a random two-dimensional subspace is shown on the left, and onto the subspace found by the algorithm on the right.

Let A_1 and A_2 equal the identity matrix, except for the two first diagonal entries, given by $A_1(1, 1) = 2$, $A_1(2, 2) = 0.3$, $A_2(1, 1) = 0.3$ and $A_2(2, 2) = 4$ respectively. We draw a random orthogonal matrix Q, and set the covariance matrices to be $\Sigma_1 = QA_1Q'$ and $\Sigma_2 = QA_2Q'$. Hence, both populations look very similar in all but two dimensions. Figure 6 shows how the points would appear projected onto the plane, had we not rotated the covariance matrices.

We apply our algorithm in a nested fashion: Given the training set, we first reduce from 10

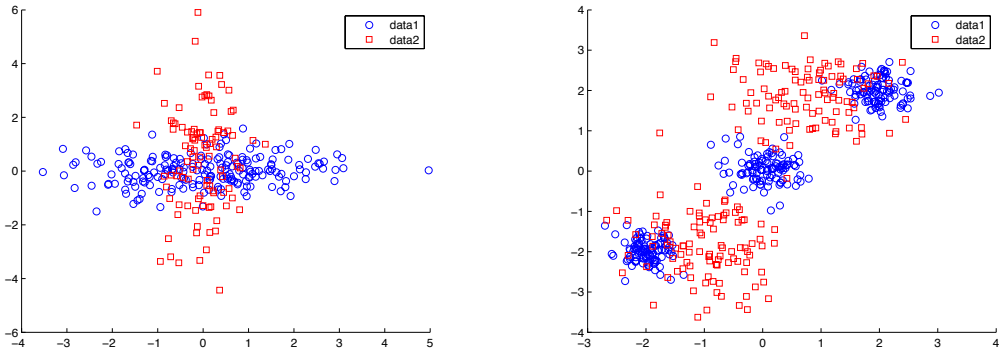

Figure 6: Points of the two populations projected onto the plane where the covariance matrices differ.

to 5 dimensions. Once the 5-dimensional subspace is found, we project the training set onto this hyperplane, and run our algorithm again, to reduce to the final number of dimensions.

We vary this to be 3, 2 and 1. Once we have projected our data from 10 dimensions to 5 and from 5 to either 3, 2 or 1, we estimate the density of both classes, and, via Bayes', compute for each observation the posterior probability of belonging to either class. We then assign each observation to the class with the highest posterior probability. In our restults, we refer to these three outputs as RFE–1, RFE–2 and RFE–3.

We present two results. First, we use the whole dataset to train the classifier, and then apply it to the entire dataset. We evaluate its performance classifying the whole set by giving the accuracy, that is, the percentage of samples correctly classified. Secondly, we perform a 5-fold cross validation, where we split the training set into 5 roughly equal parts, keep one part as the test set, and use the other four to train the classifier. Afterwards, we apply the classifier to the test set. Each observation will act as an unseen example only once, and, at the end, we obtain the proportion of correctly classified samples. Since the accuracy given in cross validation depends on the random partition of the data set into training and test sets, we carry out this procedure 20 times and report the average accuracy, as well as the standard deviation.

This analysis is also carried out with several classifiers: Linear and quadratic discriminant analysis, Naive Bayes and Support Vector Machines, as well as some variants of these. We use the corresponding built-in functions in MATLAB 2012a.

Method	Accuracy	Acc 5-fold cv (std)
LDA	55.33	51.92 (1.67)
QDA	83	76.87 (1.24)
NBGau	74.67	71.82 (1.02)
NBKer	75.67	70.67 (1.1)
SVMLin	55.67	52.3 (2.18)
SVMPoly	100	67.467 (1.77)
SVMRBF	100	69.1 (0.77)
RFE-1	80.28 (7.78)	75.37 (3.89)
RFE-2	84.92 (4.33)	81.32 (2.96)
RFE-3	86.1 (0.73)	82.73 (1.29)

From the results, we can see that when the whole set is used for training, SVMs yield the highest accuracy, with our method and QDA far behind. However, when carrying out cross validation, we see that our algorithm outperforms the rest of the methods in accuracy, when the final dimension we reduce to is either 2 or 3. When it is 1, too much information is lost and we can't differentiate the samples as well as QDA or Naive Bayes can.

The second table displays the values for the function M. For this analysis, we compare our method to those who output a posterior probability of each sample belonging to each one of the two classes. These methods are LDA, QDA and the Naive Bayes classifiers. First, we train each classifier with the whole data set, compute the posterior probability of each

observation belonging to each group, and compute M as

$$M = \frac{1}{N} \sum_j \sum_{x_i \in C_j} \log\left(P_j(x_i)\right)$$

Then we perform a 5-fold cross validation, and compute a final M as the sum of the M's computed from each one of the test sets, for that chosen partition of the data set. We repeat the procedure 20 times, and report the average of the final M's, as well as their standard deviation. We also show the mean squared error (MSE) for both scenarios, where

$$MSE = \frac{1}{N} \sum_j \sum_{x_i \in C_j} \left(1 - P_j(x_i)\right)^2$$

Method	M	M 5-fold CV (std)	MSE	MSE 5-fold CV (std)
LDA	-0.68	-0.72 (0.01)	0.24	0.26 (0.004)
QDA	-0.34	-0.45 (0.02)	0.11	0.15 (0.004)
NBGau	-0.52	-0.59 (0.01)	0.18	0.2 (0.005)
NBKer	-0.47	-0.62 (0.01)	0.15	0.21 (0.005)
RFE-1	-0.46 (0.068)	-0.5 (0.03)	0.15 (0.029)	0.16 (0.011)
RFE-2	-0.35 (0.046)	-0.41 (0.03)	0.11 (0.013)	0.14 (0.011)
RFE-3	-0.32 (0.019)	-0.39 (0.02)	0.1 (0.005)	0.13 (0.007)

Once again we see that our method outperforms the others when we project to 3 dimensions.

Using M or MSE as measures of achievement of a classifier is more revealing than the more widely used accuracy, since they do not just count the number of hits and misses as does the latter, but quantify instead the degree of confidence assigned to each classification though the posterior probability P.

For the second experiment, we produce a dataset consisting of two populations, the first one with 300 and the second one with 200 data points. Again, our observations will be 10–dimensional, and the difference between both populations will lie in a two dimensional plane. Both populations will consist of mixtures of gaussians. The first one will consist of three components and the second one of two. The means of the first mixture will be given by $(-2, -2)$, (0.0) and $(2, 2)$, and the covariance matrices will be the diagonal, with $\Sigma_{(1,1)} = \Sigma_{(2,2)} = 0.1$. The second mixture will have means $(-1, -2)$ and $(1, 2)$, and diagonal covariance matrices with $\Sigma_{(1,1)} = \Sigma_{(2,2)} = 0.5$. We show the dataset projected onto the plane of interest in Figure 6. The remaining eight entries in both classes will be draws of a standard gaussian.

Method	Accuracy	Acc 5-fold cv
LDA	54.6	46.64 (1.95)
QDA	81.6	77.21 (0.89)
NBGau	64.4	60.14 (1.13)
NBKer	70.4	59.87 (0.95)
SVMLin	54.2	46.97 (1.81)
SVMRBF	99.6	70.14 (1.34)
RFE-1	72.46 (10.56)	71.03 (4.84)
RFE-2	81.95 (4.63)	77.44 (2.97)
RFE-3	83.31 (0.33)	80.25 (0.95)

Again our method yields a higher accuracy in cross validation when the final target dimension in either 2 or 3

Method	M	M 5-fold CV	MSE	MSE 5-fold CV
LDA	-0.689	-0.7144 (0.0053)	0.2476	0.2604 (0.0026)
QDA	-0.3963	-4925 (0.0089)	0.1292	0.1624 (0.0027)
NBGau	- 0.6383	-0.6857 (0.0088)	0.2233	0.2444 (0.0039)
NBKer	-0.5798	-0.7154 (0.0108)	0.1975	0.2508 (0.0033)
RFE-1	-0.5172 (0.1026)	-0.5398 (0.049)	0.1752 (0.0428)	0.1832 (0.0203)
RFE-2	-0.4093 (0.0494)	-0.4777 (0.0279)	0.1322 (0.0201)	0.1569 (0.0115)
RFE-3	-0.3906 (0.0060)	-0.4507 (0.0091)	0.1249 (0.0021)	0.1459 (0.0031)

6 Application

We apply our algorithm to the Wisconsin Diagnostic Breast Cancer classification problem, available in the Machine Learning Repository of the University of California Irvine. We decided to use this data set because several groups have applied different classification methods to it. A detailed description of the data set can be found in [8]. The dataset consists of 30 features extracted from images of biopsies of the breast, such as symmetry, perimeter, etc. There are 569 images, corresponding to 212 malignant and 357 benign cases.

6.1 Data and preprocessing

We normalize the data so that each each feature has mean zero and variance one across all the patients. Also, we can toss out uninformative features as follows: Our algorithm chooses hyperplanes in which, after estimating the density of each population as a Gaussian, we can assign a posterior probability of each observation belonging to the right population the most accurately. Hence, we can compute our function M for each feature, which is a fast to compute, one-dimensional estimate, and then choose a small number of the features that scored a high value of M. In our case, we kept the 20 features with the highest M, in descending order.

6.2 Methodology

We apply our algorithm in a nested fashion: Given the training set, we first reduce from 20 to 10 dimensions. Once the 10-dimensional subspace is found, we project the training set onto this hyperplane, and run our algorithm again, to reduce to 5 dimensions. Having found an optimal 5-dimensional hyperplane, we once more project the 10 dimensional data onto the 5 dimensional hyperplane, and one last time, apply our method to reduce from 5 dimensions to one.

Once we have projected our data from 20 dimensions to 10, from 10 to 5 and from 5 to 1, we estimate the density of both classes and, via Bayes', compute for each observation the posterior probability of belonging to either class. We then assign each observation to the class with the highest posterior probability.

As with the synthetic examples of the prior section, we calculate the accuracy of the classifier in sample and through 10 and 5-fold cross-validation.

Method	Accuracy	Acc 10-fold CV (std)	Acc 5-fold cv (std)
LDA	96.84	96.07 (0.24)	96.05 (0.31)
QDA	97.54	95.6 (0.23)	95.55 (0.32)
NBGau	94.02	93.33 (0.29)	93.36 (0.29)
NBKer	95.08	94.37 (0.22)	94.18 (0.29)
SVMLin	98.95	97.22 (0.33)	97.12 (0.43)
SVMQuad	100	94.09 (0.69)	93.55 (0.79)
SVMPoly	100	94.17(0.44)	94.48 (0.52)
SVMRBF	100	90.41 (1.04)	88.61 (1.45)
RFE–1	97.92 (0.3167)	97.12 (0.54)	96.86 (0.96)
RFE–2	97.69 (0.5077)	96.74 (0.5)	96.74 (0.53)
RFE–3	97.72 (0.4365)	96.43 (0.52)	96.34 (0.61)

Applying our model in sample, using the whole data set first to train and then to test, we obtained 560 correctly classified samples out of 569. The SVM models yielded a higher accuracy. However, in the 10-fold cross validation, our method gave the highest accuracy.

Again as with the synthetic examples before, the second table displays the values for the function M and the mean squared error MSR in and out of sample for those methods that output a posterior probability of each sample belonging to each one of the two classes.

Method	M	M 10-fold CV (std)	MSE	MSE 10-fold CV (std)
LDA	-0.0783	-0.1118 (0.0045)	0.0225	0.0291 (0.0011)
QDA	-0.2548	-0.5873 (0.0500)	0.0254	0.0400 (0.0018)
NBGau	-0.5369	-0.6587 (0.0691)	0.0580	0.0629 (0.0017)
NBKer	-0.3821	-0.5551 (0.1235)	0.0430	0.0519 (0.0018)
RFE–1	-0.0778 (0.005)	-0.1022 (0.0183)	0.019 (0.0016)	0.0262 (0.0060)
RFE–2	-0.0754 (0.0085)	-0.1030 (0.0127)	0.0176 (0.0021)	0.0258 (0.0029)
RFE–3	-0.0690 (0.008)	-0.1070 (0.0174)	0.0163 (0.002)	0.0266 (0.0028)

The methodology of this article, when projecting to 1 dimension, obtains an accuracy comparable to linear SVM's in the cross-validation scenarios. It is also the method that minimizes both the mean squared error and that maximizes our function M.

Figure 7 displays the observations projected onto the optimal two dimensional subspace found by our algorithm.

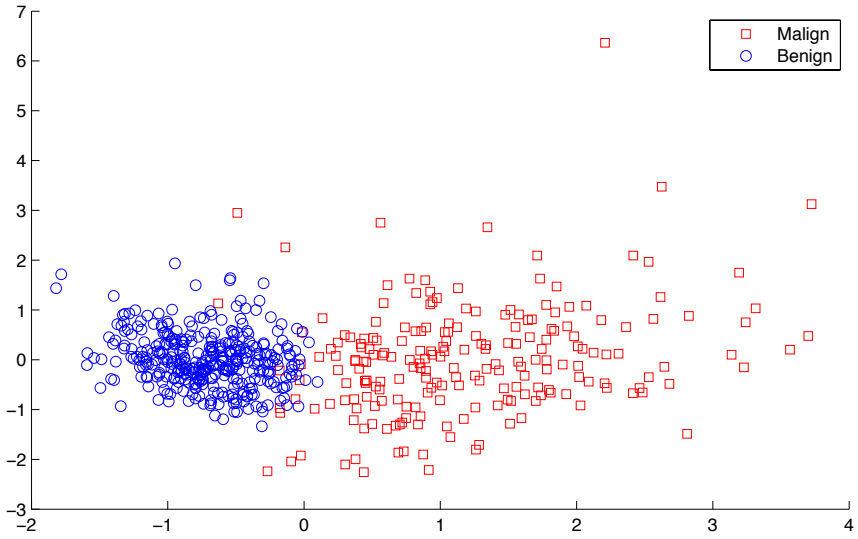

Figure 7: Projection of the data onto the optimal 2-dimensional plane. The algorithm finds a plane where the benign cases are tightly clustered together. One can also notice that a Gaussian model fits well both populations, and that they are far apart and different enough that classifying using the posterior probability from Bayes theorem will yield good results.

7 Conclusions

A novel method for finding a lower dimensional representation of high dimensional observations corresponding to different classes has been developed. The methodology proceeds through small rotations of the subspace sought. For a given subspace, the probability density of the various classes is estimated, and used through Bayes to find a posterior probability that each observation belongs to each class. The objective function guiding the rotations is the Kullback-Leibler divergence between this soft assignment and the true membership of the data.

The methodology has been tried on two synthetic examples and to the WBCD dataset. In all cases, it outperforms several benchmark methods; for the latest example, it slightly underperforms linear support vector machines, though the latter only provides a hard assignment,

with no probability associated with it.

Appendix: Computing $\frac{\partial \rho(z)}{\partial \theta}$ efficiently

Let

$$\frac{\partial \rho_{icl}(z)}{\partial \theta}, \qquad \theta = S^i_j$$

denote the derivative with respect to the rotation angle θ in the plane $i-j$, of the density ρ, with parameters μ and Σ computed using the population (icl), evaluated at a point z. We have three indices: two for the plane and one for the population from which ρ was computed. Hence,

$$\begin{aligned}\frac{\partial \rho_{icl}(z)}{\partial \theta} = \frac{\rho_{icl}(z)}{2}[&(z-\mu_{icl})^\top(\Sigma^{-1}_{icl}\Sigma_{\theta,icl}\Sigma^{-1}_{icl})(z-\mu_{icl}) \\ &- (z-\mu_{icl})^\top\Sigma^{-1}_{icl}(z_\theta-\mu_{\theta,icl}) - (z_\theta-\mu_{\theta,icl})^\top\Sigma^{-1}_{icl}(z-\mu_{icl}) \\ &- \mathrm{tr}(\Sigma^{-1}_{icl}\Sigma_{\theta,icl})]\end{aligned}$$

In order to examine the different components of this expression, let us define

$$\begin{aligned}B_1 &= (z-\mu_{icl})^\top(\Sigma^{-1}_{icl}\Sigma_{\theta,icl}\Sigma^{-1}_{icl})(z-\mu_{icl}) \\ B_2 &= (z-\mu_{icl})^\top\Sigma^{-1}_{icl}(z_\theta-\mu_{\theta,icl}) \\ B_2' &= (z_\theta-\mu_{\theta,icl})^\top\Sigma^{-1}_{icl}(z-\mu_{icl}) \\ B_3 &= \mathrm{tr}(\Sigma^{-1}_{icl}\Sigma_{\theta,icl}),\end{aligned}$$

so that

$$\frac{\partial \rho_{icl}(z)}{\partial \theta} = \frac{\rho_{icl}(z)}{2}\left(B_1 - B_2 - B_2' - B_3\right).$$

- B_1: Let $v := (z-\mu_{icl})^\top\Sigma^{-1}_{icl}$, so $B_1 = v\Sigma_{\theta,icl}v$. The matrix $\Sigma_{\theta,icl}$'s only non-zero entries are at the i-th column and row, with values

  $$u := \overline{(z_\theta(i)-\mu_{\theta,icl}(i))\,(z-\mu_{icl})},$$

  where the bar denotes averaging over all observations z in the icl class. Then

  $$B_1 = 2v(i) < u, v > .$$

- B_2 and B_2': Since the vector $(z_{\theta,jcl}-\mu_{\theta,icl})$ has its only non-zero entry at the i-th place, with value $(z_{\theta,jcl}(i)-\mu_{\theta,icl}(i))$, in order to compute B_2 we just need to compute the following product

  $$B_2 = (z_{\theta,jcl}(i)-\mu_{\theta,icl}(i))\left\langle (z_{jcl}-\mu_{icl}), \Sigma^{-1}_{icl}(i_{\text{th row}})\right\rangle.$$

  Since Σ^{-1} is symmetric, B_2' is equal to B_2.

- B_3: To compute this term, we only need to add the diagonal entries of a matrix times a matrix whose only non-zero entries are given in one column and one row. Hence, we have

  $$B_3 = 2\left\langle u, \Sigma^{-1}_{icl}(i_{\text{th row}})\right\rangle$$

References

[1] Bishop, C. M., *Pattern recognition and machine learning*, Springer, 2006.

[2] I. Rish., "An empirical study of the Naive Bayes Classifier", *In Proceedings of IJCAI-01 workshop on Empirical Methods in AI*, 41–46, 2001.

[3] Mika, S., Rätsch, G., Weston, J., Schölkopf, B., and Müller, K. R. (1999). Fisher discriminant analysis with kernels. In Proc. IEEE Neural Networks for Signal Processing Workshop, NNSP.

[4] Guyon, I. and Elisseeff, A., "An Introduction to Variable and Feature Selection", *Journal of Machine Learning Research 3*, 1157-1182, 2003.

[5] Kullback, S. and Leibler, R. A., "On information and sufficiency", *The Annals of Mathematical Statistics*, **22**, 79–86, 1951.

[6] Hastie, T. and Tibshirani, T., "Discriminant Analysis by Gaussian mixtures", *Journal of the Royal Statistical Society. Series B*, **58**, 155–176, 1996.

[7] Jain, A. K., Duin, R. P. W. and Mao, J., "Statistical Pattern Recognition: A Review", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**, No. 1, 2000.

[8] Sewak, M., Vaidya, P., Chan, C.C. and Duan Z.H.,"SVM approach to Breast Cancer Classification", *Second International Multisymposium on Computer and Computational Science, IEEE"*, 32–37, 2007.

[9] J. Han, M. Kamber, *Data Mining*, Morgan Kaufmann Publishers, 2001.