Iterative Supervised Principal Components

Juho Piironen
juho.piironen@aalto.fi

Aki Vehtari
aki.vehtari@aalto.fi

Helsinki Institute for Information Technology, HIIT
Department of Computer Science, Aalto University

Abstract

In high-dimensional prediction problems, where the number of features may greatly exceed the number of training instances, fully Bayesian approach with a sparsifying prior is known to produce good results but is computationally challenging. To alleviate this computational burden, we propose to use a preprocessing step where we first apply a dimension reduction to the original data to reduce the number of features to something that is computationally conveniently handled by Bayesian methods. To do this, we propose a new dimension reduction technique, called iterative supervised principal components (ISPCs), which combines variable screening and dimension reduction and can be considered as an extension to the existing technique of supervised principal components (SPCs). Our empirical evaluations confirm that, although not foolproof, the proposed approach provides very good results on several microarray benchmark datasets with very affordable computation time, and it can also be very useful for visualizing high-dimensional data.

1 INTRODUCTION

Inference in high-dimensional problems, where the number of features may greatly exceed the number of training instances, remains a topic of active research. The frequentist approaches typically formulate the problem as an optimization task with a penalty that forces the solutions to be sparse, the most popular example being the Lasso (Tibshirani, 1996), but various others have also been proposed (e.g., Fan and Li, 2001; Zou and Hastie, 2005; Zou, 2006; Candes and Tao, 2007). In the Bayesian literature, the domi-

nant approach is to use a use sparsifying prior, such as the spike-and-slab (Mitchell and Beauchamp, 1988; George and McCulloch, 1993) or the horseshoe (Carvalho et al., 2010). Inference is typically carried out by using Markov chain Monte Carlo (MCMC), but also expectation-maximization (EM) based mode finding strategies have gained popularity recently (Rockova and George, 2014; Chang et al., 2016; Bhadra et al., 2017). Empirical evidence indicates that the Bayesian approach is more accurate (Polson and Scott, 2011; Piironen and Vehtari, 2017b,c; Bhadra et al., 2017) but is computationally expensive for large number of features, especially if MCMC is used for inference.

This paper studies a practical strategy for alleviating the computational burden related to the Bayesian inference in these problems via dimension reduction. We investigate the following two-step procedure. First, we perform a dimension reduction which reduces the number of features to something that is computationally conveniently handled by fully Bayesian methods. Second, we perform the Bayesian model fitting using the reduced set of features with a sparsifying prior that will discover which of these new features are the most relevant.

Although not routinely used in the Bayesian workflow, this approach is certainly not new and has actually been very successful in empirical evaluations. Most notably, this was the key idea behind the overall winners of the NIPS 2003 feature selection challenge (Neal and Zhang, 2006), who used feature screening based on univariate significance tests and dimension reduction with principal component analysis (PCA) to reduce the dimensionality of the problem. There have also been many other explorations on these ideas (with both Bayesian and non-Bayesian emphasis), such as the supervised PCA (SPCA) (Bair et al., 2006; Yu et al., 2006). Especially the various screening approaches have proved to be promising and have received attention during the recent years (Fan and Lv, 2008; Song and Liang, 2015; Mukhopadhyay and Dutta, 2016; Ahmed and Bajwa, 2017; Chen and Dunson, 2017).

We propose a new method, called iterative supervised PCA (ISPCA) that combines screening and dimension reduction

---

in such a way that the produced set of features aims to be maximally relevant for predicting the target variable. The method is most closely related to the SPCA and could be considered as an extended version of it.

The main contributions of the paper are summarized as follows. We present a (non-trivial) extension to the original SPCA method. Unlike the original formulation, our method is model independent and does not need cross-validation for estimating the screening parameter. We also show how to handle multiclass classification problems, which was not discussed by the original SPCA paper. Based on the empirical evaluation, our method is overall competitive with the PCA and SPCA, but sometimes yields better results when used for predictive model construction. When used for visualizing high-dimensional data using only a few features, our method consistently yields at least as good and sometimes considerably better results than PCA or SPCA, which makes it a useful tool for exploratory analysis.

We would like to point out that the two-step procedure discussed in this paper is not fully Bayesian as it uses the data twice: first when constructing the new feature representation and second time when fitting the predictive model. Nevertheless, we are sometimes willing to relax the full Bayesian view in the pursuit for a scalable method that allows us to handle high-dimensional problems in a computationally feasible manner.

2 BACKGROUND

This section briefly reviews some background essential for understanding our method.

2.1 Principal Components

Assume we are given dataset with feature matrix $\mathbf{X} \in \mathbb{R}^{n \times D}$ and target values $\mathbf{y} \in \mathbb{R}^n$. Throughout this paper will assume each column of $\mathbf{X}$ is standardized to have a zero mean and unit variance if not otherwise stated. In linear dimension reduction we find a transformed set of features $\mathbf{Z}$ that are typically (but not necessarily) orthogonal and obtained by a linear projection of the original feature matrix onto a set of vectors $\mathbf{W} \in \mathbb{R}^{D \times K}$

$$\mathbf{Z} = \mathbf{X}\mathbf{W}. \tag{1}$$

Principal components analysis (PCA), where $\mathbf{W}$ consists of $K \leq \min(n-1, D)$ first right singular vectors of $\mathbf{X}$, is a well-known example of such method and a natural choice for dimension reduction.

However, because PCA is an unsupervised technique, there is no guarantee that the projections onto the first K principal components would result in an informative set of features $\mathbf{Z}$ regarding the prediction of $\mathbf{y}$. For instance, suppose we would like perform linear regression of $\mathbf{y}$ onto some set of

features and $\boldsymbol{\beta}_* \in \mathbb{R}^D$ denotes the optimal coefficients in the original feature space $\mathbf{X}$. Now, if the number of features D greatly exceeds the number of training instances n and there is enough variation in $\mathbf{X}$ unrelated to $\mathbf{y}$, it is possible that $\boldsymbol{\beta}_*$ is not in the space spanned by the principal components (the column space of $\mathbf{W}$). This means it is impossible to recover the optimal solution using the transformed set of features $\mathbf{Z}$. Even if the optimal solution would be recoverable, the solution is not necessarily sparse in the new feature space (even if it was in the original space) which can make the learning more difficult and may require a large number of transformed features K.

2.2 Supervised Principal Components

Supervised PCA (SPCA) (Bair et al., 2006) is a technique to alleviate problems with the standard unsupervised PCA. SPCs are computed as follows:

1. Compute the univariate scores $s_j = S(\mathbf{x}_j, \mathbf{y})$ between each feature $\mathbf{x}_j$ and the target variable $\mathbf{y}$.

2. Retain only those features with univariate score above some threshold γ, and compute the first (or first few) principal components of the reduced feature matrix $\mathbf{X}_\gamma$.

The score function $S(\mathbf{x}_j, \mathbf{y})$ is generally taken to be the absolute univariate regression coefficient between $\mathbf{x}_j$ and $\mathbf{y}$ which is up to a constant the same as the (absolute) correlation between the two variables. For determining an appropriate threshold γ, Bair et al. (2006) proposed to use cross-validation for the final prediction model that utilizes the extracted features.

SPCA can written in the form (1) by padding the principal components of $\mathbf{X}_\gamma$ with zeros corresponding to the features that were screened out. The benefit of SPCA compared to the standard PCA is that the screening step anticipates other sources of variation in $\mathbf{X}$ unrelated to the target variable $\mathbf{y}$, and thus the extracted features will typically be more related to the relevant variation.

Some problems still persist, however. One is that the screening step ignores the uncertainty about the relevance of the features with univariate score s_j less than γ. Although for many datasets this does not appear to be harmful from predictive point of view, we would like a more principled approach for treating the remaining features than simply ignoring them since it is possible for a feature to be relevant even if its univariate score would be exactly zero (see example in Sec. 4.1). Secondly, choosing the thresholding parameter via cross-validation makes the construction of SPCs dependent of the model used for prediction. This can make the procedure computationally expensive (especially if Bayesian model is used) and it would be conceptually more satisfactory to find model independent procedure for

the dimension reduction. A third issue is that the original formulation of Bair et al. (2006) does not provide a way of handling classification problems with more than two class. The next section discusses our proposed method that is inspired by the idea of SPCA but aims to provide a solution for all these problems.

3 ITERATIVE SUPERVISED PRINCIPAL COMPONENTS

This section discusses our proposed method of iterative supervised PCA (ISPCA). We shall first outline the algorithm and then discuss its properties, further ideas and implementational details in more detail.

3.1 Outline of the Algorithm

The algorithm consists of iterating the following steps K times:

1. Compute the univariate scores $s_j = S(\mathbf{x}_j, \mathbf{y})$ for each feature $\mathbf{x}_j$.

2. Retain only features with univariate score $s_j > \gamma$, and compute the first principal component $\mathbf{v}_\gamma$ of these features $\mathbf{X}_\gamma$. Choose γ so that the projection of $\mathbf{X}_\gamma$ onto this vector $\mathbf{z}_\gamma = \mathbf{X}_\gamma \mathbf{v}_\gamma$ maximises the score $S(\mathbf{z}_\gamma, \mathbf{y})$. Denote the extracted feature by $\mathbf{z}$.

3. Subtract the variation explained by $\mathbf{z}$ from each column in $\mathbf{X}$ (including those that were screened out at step 2) as $\mathbf{x}'_j = \mathbf{x}_j - b_j \mathbf{z}$ where $b_j = (\mathbf{z}^\mathsf{T} \mathbf{z})^{-1}(\mathbf{x}_j^\mathsf{T} \mathbf{z})$. This yields a modified feature matrix $\mathbf{X}'$.

4. Set $\mathbf{X} \leftarrow \mathbf{X}'$ and go to step 1.

The intuition behind the algorithm is as follows. At step 2 of each iteration, we seek direction that is maximally relevant for explaining variance of $\mathbf{y}$. Step 3 ensures that the subsequent directions will capture variation that is not explained by the directions that we have computed so far. This is useful, because there may be features in $\mathbf{X}$ that are screened out at step 2 but are still correlated with those that are retained after screening, and we do not want subsequent latent features $\mathbf{z}$ to be correlated (see discussion below).

3.2 Properties of the Method

Like PCA and SPCA, the algorithm in Section 3.1 results in a transformed set of features $\mathbf{Z} \in \mathbb{R}^{n \times K}$ (computed at step 2) that are orthogonal and obtained by a linear projection of the original feature matrix onto a set of vectors $\mathbf{W} \in \mathbb{R}^{D \times K}$ as in Equation (1).

To see that the features $\mathbf{Z}$ will be orthogonal, consider the following. After step 3 in the algorithm, all the columns of

$\mathbf{X}'$ will be orthogonal to $\mathbf{z}$ because

$$
\begin{aligned}
\mathbf{x}'^\mathsf{T}_j \mathbf{z} &= (\mathbf{x}_j - b_j \mathbf{z})^\mathsf{T} \mathbf{z} \\
&= \left(\mathbf{x}_j - (\mathbf{z}^\mathsf{T} \mathbf{z})^{-1}(\mathbf{x}_j^\mathsf{T} \mathbf{z}) \mathbf{z} \right)^\mathsf{T} \mathbf{z} \\
&= \mathbf{x}_j^\mathsf{T} \mathbf{z} - (\mathbf{z}^\mathsf{T} \mathbf{z})^{-1}(\mathbf{x}_j^\mathsf{T} \mathbf{z}) \mathbf{z}^\mathsf{T} \mathbf{z} \\
&= 0,
\end{aligned}
$$

and therefore any linear combination of these (that is, the latent feature to be extracted at the next iteration) will also be orthogonal to $\mathbf{z}$. Using induction, it is straightforward to show that each latent feature is orthogonal also to all the other extracted features, not only to the previous one (the proof is omitted).

To prove that the extracted features can be written in the form of Equation (1), we need to set up some notation. Denote the feature matrix used at steps 1 and 2 at iteration k as $\mathbf{X}_k$, so that $\mathbf{X}_1 = \mathbf{X}$ is the original feature matrix. Moreover, denote the principal components computed at step 2 as $\mathbf{v}_1, \ldots, \mathbf{v}_K$. For notational convenience, we shall now assume that these vectors are padded with zeros corresponding to those features that were screened out at the corresponding iteration, so that each $\mathbf{v}_k \in \mathbb{R}^D$.

Using this notation, the latent variables $\mathbf{z}_k$ computed at step 2 satisfy $\mathbf{z}_k = \mathbf{X}_k \mathbf{v}_k$ for all $k = 1, \ldots, K$. The construction of the next feature matrix $\mathbf{X}_{k+1}$ from the previous one $\mathbf{X}_k$ at step 3 can be written in a matrix form as

$$
\mathbf{X}_{k+1} = \mathbf{X}_k - \mathbf{Z}_k \mathbf{B}_k, \tag{2}
$$

where all the columns of $\mathbf{Z}_k \in \mathbb{R}^{n \times D}$ are equal to $\mathbf{z}_k$, and $\mathbf{B}_k$ is a diagonal matrix with elements $b_1, \ldots, b_D$ from iteration k. We can rewrite $\mathbf{Z}_k = \mathbf{X}_k \mathbf{V}_k$ where $\mathbf{V}_k \in \mathbb{R}^{D \times D}$ with all columns equal to $\mathbf{v}_k$. By plugging this into (2) we get

$$
\begin{aligned}
\mathbf{X}_{k+1} &= \mathbf{X}_k - \mathbf{X}_k \mathbf{V}_k \mathbf{B}_k \\
&= \mathbf{X}_k (\mathbf{I} - \mathbf{V}_k \mathbf{B}_k) \quad | \quad \mathbf{A}_k := \mathbf{I} - \mathbf{V}_k \mathbf{B}_k \\
&= \mathbf{X}_k \mathbf{A}_k,
\end{aligned}
$$

from which we deduce

$$
\mathbf{X}_k = \mathbf{X} \mathbf{A}_1 \mathbf{A}_2 \ldots \mathbf{A}_{k-1} = \mathbf{X} \prod_{t=1}^{k-1} \mathbf{A}_t.
$$

This lets us write the latent features $\mathbf{z}_k$ as

$$
\mathbf{z}_k = \mathbf{X}_k \mathbf{v}_k = \mathbf{X} \left(\prod_{t=1}^{k-1} \mathbf{A}_t \right) \mathbf{v}_k,
$$

and thereby we arrive at decomposition (1) by defining the columns of the projection matrix $\mathbf{W}$ as

$$
\mathbf{w}_k = \left(\prod_{t=1}^{k-1} \mathbf{A}_t \right) \mathbf{v}_k = \left(\prod_{t=1}^{k-1} (\mathbf{I} - \mathbf{V}_t \mathbf{B}_t) \right) \mathbf{v}_k. \tag{3}
$$

In practice we never form matrices $\mathbf{V}_k$ or $\mathbf{B}_k$ to compute $\mathbf{W}$. By exploiting the structure of these matrices, the columns of $\mathbf{W}$ can be computed much more efficiently (see details in the supplementary material).

It is worth noticing that although the new features $\mathbf{Z}$ will be orthogonal, the columns of the rotation matrix $\mathbf{W}$ typically will not. This is not a handicap and can, in fact, be very beneficial as it allows detecting features that are not relevant alone but become relevant after some other features are included in the model (see Sec. 4.1 for a simple example).

3.3 Combination of Supervised and Unsupervised Components

In principle, we could extract $\min(n-1, D)$ ISPCs from the data (or until none of the features have univariate score numerically distinguishable from zero). In practice, however, this is not advisable and we call this the *naive* algorithm. This is because the process of repeatedly finding the most relevant direction can overfit especially when the sample size n is small because some features may have a relatively large absolute sample correlation with $\mathbf{y}$ although they are completely irrelevant, simply due to random fluctuation in the data. Thus the algorithm may find "relevant" features that are in fact noise. This will result in biases in the inference when the extracted features are later used for visualization or predictive model construction. Thus in practice we typically extract only a few supervised components, and if needed, compute the standard unsupervised PCs with the rest of the data variation. A practical automatic strategy for deciding the number of supervised components is discussed in Section 3.4.

After the supervised iteration, we can compute standard principal components as usual but with the exception that these are now computed from the modified data matrix $\mathbf{X}'$ that we are left with after the supervised iteration (after subtracting the variation explained by the K supervised components at step 3 of each iteration). If we denote the total number of components by K_{tot} and the unsupervised components by $\mathbf{v}_k,\ k = K+1,\ldots,K_{\text{tot}}$, the columns of the final projection matrix $\mathbf{W} \in \mathbb{R}^{D \times K_{\text{tot}}}$ corresponding to the unsupervised components are given by

$$\mathbf{w}_k = \left(\prod_{t=1}^{K}(\mathbf{I} - \mathbf{V}_t\mathbf{B}_t) \right) \mathbf{v}_k, \quad k = K+1,\ldots,K_{\text{tot}}.$$
$$(4)$$

It is worth noticing that after this process, all the extracted features $\mathbf{z}_k$ (both supervised and unsupervised) will be orthogonal, which is often useful. The inclusion of unsupervised components can be important for constructing a good predictive model. This is simply due to the fact that not always all the relevant variation will be captured by the first supervised components. This point will be demonstrated experimentally in Section 4. We also point out that we can

apply this same idea for the original SPCA, that is, compute unsupervised components from the features screened out, and make these unsupervised features orthogonal to the supervised ones.

3.4 Deciding the Number of Supervised Components

As discussed in Section 3.3, the unrestricted supervised iteration may overfit, that is, find features that are appear relevant but are in fact noise. Fortunately, there is a simple but effective way of discovering how many components we can extract without substantial overfitting. We do this using a permutation test. Before computing the next supervised principal component at step 2 of each iteration, we compute a p-value

$$p = \frac{1}{R} \sum_{r=1}^{R} \mathbb{1}\left(\max_j S(\mathbf{x}_j, \mathbf{y}_r) \geq \max_j S(\mathbf{x}_j, \mathbf{y}) \right), \quad (5)$$

where $\mathbf{y}_r$ denotes a random permutation of the original $\mathbf{y}$ and $\mathbb{1}(E) = 1$ if event E is true and zero otherwise. Quantity (5) estimates how likely it is that the largest univariate score would be as extreme as actually observed if none of the variables $\mathbf{x}_j$ were actually related to $\mathbf{y}$. If $p < \alpha$ for a relatively small α, we have strong evidence that there is still relevant variation left in the data and we can extract the next component being fairly confident that the finding was not a false discovery. If $p \geq \alpha$, we stop the supervised iteration and proceed to extracting unsupervised features if needed.

In our experiments we used $\alpha = 0.01$ and $R = 1000$ random permutations, which makes the number of false discoveries small. In principle we believe that it is better to be too conservative in setting α than to allow the algorithm to overfit. After all, for predictive model construction we can always compute the standard unsupervised PCs with the rest of the data variation, use Bayesian model with a sparsifying prior and let the data decide which components are really relevant and which not. The results indicate that this strategy is both computationally feasible and performs well in practice.

3.5 More Algorithmic Details

Finding the optimal screening threshold γ at step 2 would require computing the first principal component for all feature subset sizes from 1 to D which is computationally expensive. In practice we use a more crude search and set up an evenly spaced grid of values between $\gamma_{\min}$ and $\gamma_{\max}$, where $\gamma_{\max}$ is the smallest γ so that all but one feature are screened out, and $\gamma_{\min}$ the largest γ so that the number of features after screening is W. We could set $W = D$ but since in practice the optimal γ is rarely so that almost all features survive the screening, we typically use $W < D$ which makes the algorithm faster and concentrates the grid on more plausible values. The computational complexity of computing a PC

among at most W features is the minimum of $O(WN^2)$ and $O(W^2N)$, which shows that computational savings can be obtained by adjusting the feature window size W. In our experiments we used grid of size $M = 10$ with feature window limit $W = 500$ which seem to provide good balance between accuracy and speed. In fact, especially when n is fairly large, typically more time is spent in the permutation test (Sec. 3.4) which scales as $O(RND)$.

As a minor detail, we mention that before using the new features $\mathbf{Z}$ for predictive model construction, we typically normalize them to have unit variance so that none of the features is favored a priori. For visualization purposes this is not necessary but does not hurt either.

3.6 Multiclass Classification and Other Observation Models

The supervised algorithm in Section 3.1 can naturally be extended to classification problems with $C > 2$ classes. We do this by defining C binary variables $\mathbf{y}_c = \mathbb{1}(\mathbf{y} = c)$, that is, "class c or some other class", for $c = 1, \ldots, C$. We then repeat steps 1 and 2 for all these C auxiliary target variables which yields candidate directions $\mathbf{v}_c$, $c = 1, \ldots, C$ from which then choose the one which maximizes the score $S(\mathbf{X}\mathbf{v}_c, \mathbf{y}_c)$. This typically results in direction $\mathbf{v}$ that tries to separate one of the classes from the rest (see the multiclass example in Sec. 4.2).

We propose to use this same idea also for SPCA; in this case we define the univariate scores in the screening to be the maxima of the C scores as $\tilde{S}(\mathbf{x}_j, \mathbf{y}) = \max_c S(\mathbf{x}_j, \mathbf{y}_c)$. Although simple, this approach turns out to be quite successful, and lets us extend also SPCA to multiclass problems (not discussed by Bair et al. (2006)).

When computing the univariate scores $S(\mathbf{x}_j, \mathbf{y})$ in other than regression or classification problems we could use pseudo-data $\mathbf{t}$ in place of $\mathbf{y}$, so that $\mathbf{t}$ is derived from the second order expansion to the log-likelihood from an univariate (generalized) regression of $\mathbf{y}$ onto $\mathbf{x}_j$. This is discussed by Bair et al. (2006) so we do not discuss it further here.

3.7 Interpretation and Obtaining a Sparse Solution in the Original Space

The columns of the projection matrix $\mathbf{W}$ are directly interpretable by investigating which entries are nonzero, as the corresponding features are likely to be correlated and predictive about $\mathbf{y}$ (at least if the column was computed in a supervised fashion). If a linear model is used with the new set of features $\mathbf{Z}$, the corresponding regression coefficients $\tilde{\boldsymbol{\beta}}$ can be transformed back to the original feature space simply as $\boldsymbol{\beta} = \mathbf{W}\tilde{\boldsymbol{\beta}}$.

Although the columns of $\mathbf{W}$ will typically have a lot of zeros, they can still contain quite a few nonzeros as the nonzero entries correspond to correlated features that carry
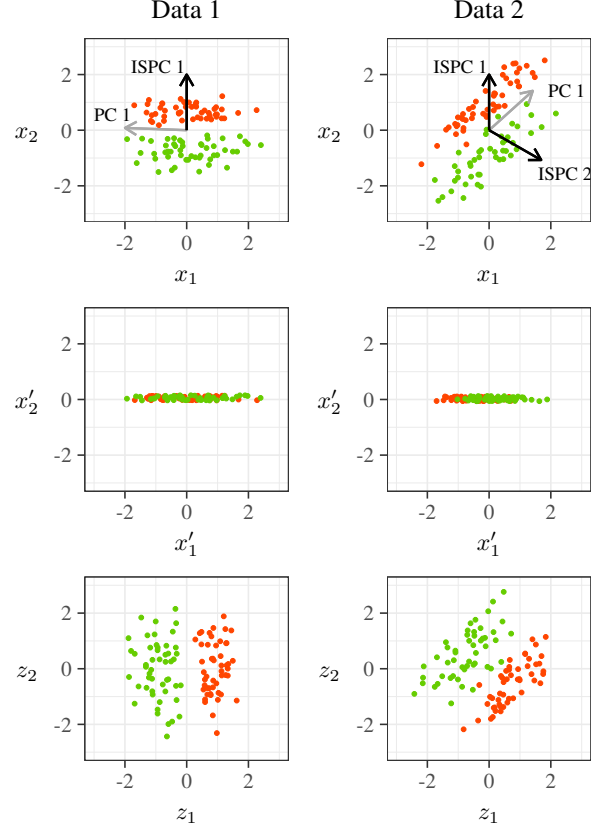

Figure 1: Illustration of ISPCA for two toy binary classification datasets (the two columns). Top row shows the original dataset (colors denoting the different classes), supervised components found by ISPCA with the permutation test, and the first unsupervised PC. Middle row shows the feature matrix obtained after subtracting the variation related to the first ISPC from $\mathbf{X}$ (in the plots, x_2' is exactly zero but has been jittered by a small amount to aid visualization). Bottom row shows the transformed features $\mathbf{Z}$.

similar information. To obtain an even more sparse solution in the original feature space, we can use the projective variable selection framework (Goutis and Robert, 1998; Dupuis and Robert, 2003) which has shown to be successful for finding a sparse solution when there is redundancy in the features (Piironen and Vehtari, 2017a). This technique has also been studied from a non-Bayesian viewpoint with good results, and is known as "preconditioning" for variable selection (Paul et al., 2008). Due to the space constraints, we do not discuss this further but merely point out that this is possible.

4 EXPERIMENTS

4.1 Toy Examples

We first illustrate the use of ISPCA with two simple toy problems that will shed light on the algorithm, see Figure 1.
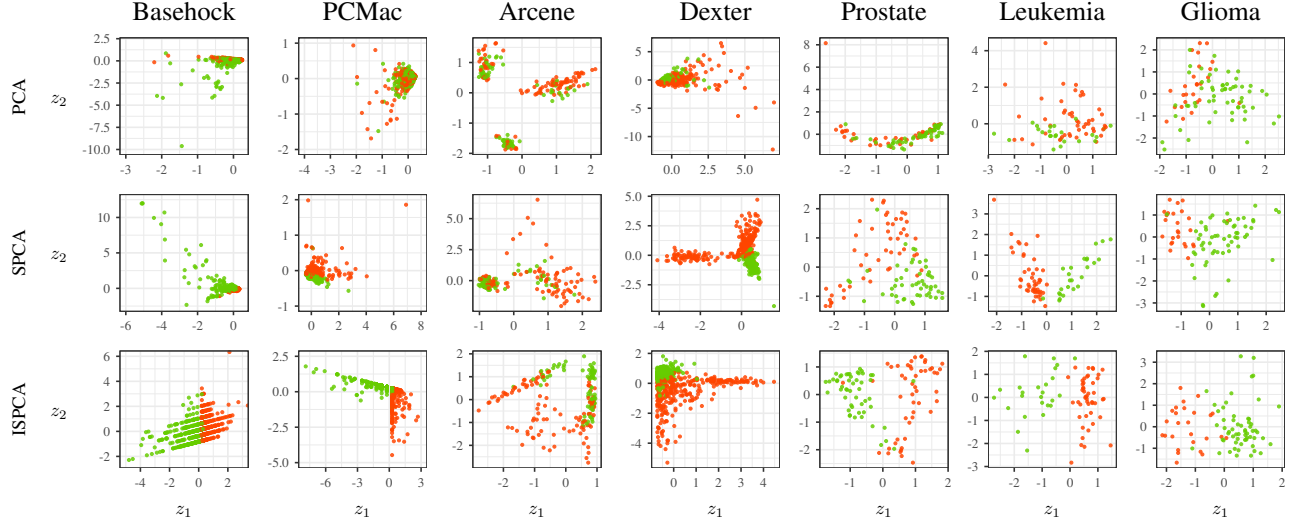
Figure 2: Some of the binary classification datasets visualized using first two latent features obtained by PCA (top row), SPCA (middle row) and ISPCA (bottom row). Colors refer to the two classes. For Basehock and PCMac the visualization is done using only a subset of the data to reduce data overlap, but the features are extracted using the full datasets. For the last three datasets, the second feature of ISPCA is actually unsupervised, since only one supervised component was supported by the data.

The first column shows data where only one of the variables (x_2) is relevant for separating the two classes. Out of the two variables, x_2 has higher univariate score, and since this is higher than the univariate score for the principal component of the two features (x_1, x_2), the first ISPC points to direction $\mathbf{w} = (0, 1)$. After subtracting the variation explained by this direction from the feature matrix $\mathbf{X}$, we end up with a modified feature matrix $\mathbf{X}'$ where only x_1' has nonzero variance (middle row). However, this feature has univariate score close to zero (thereby failing the permutation test), and hence the supervised iteration terminates. If we compute unsupervised PCA using this rest of the data variation $\mathbf{X}'$, we end up with transformed features (z_1, z_2) (bottom row), where only the first one is supervised and also the only relevant feature.

The second column shows a more interesting example. Again, feature x_1 is irrelevant alone (has univariate score close to zero), but becomes relevant together with x_2 (that is, x_1 and x_2 together have better class separation than x_2 alone). Again the first ISPC points towards $\mathbf{w} = (0, 1)$, but now after subtracting the variance explained by this direction from $\mathbf{X}$, since x_1 and x_2 are correlated, we end up with a new feature matrix $\mathbf{X}'$ where the feature x_1' has a significant correlation with the class label. The first PC of $\mathbf{X}'$ points to direction $\mathbf{v} = (0, 1)$, but transforming this back to the original feature space using Equation (3), the second ISPC points roughly to direction $\mathbf{w} = (1.6, -1)$ in the original space (top plot). This results in a new set of features (z_1, z_2) out of which both are supervised and about equally relevant (bottom plot).

In both of these examples the first unsupervised PC does

not explain variation relevant for separating the two classes. SPCA would work well in the first case because then x_1 would be screened out and the first SPC would be equal to the first ISPC. However, the second case shows an example where ISPCA has a distinctive advantage over the SPCA. Also in this case SPCA would screen x_1 out and would find only the first relevant direction (that is, feature x_2), whereas the iterative procedure can discover that x_1 becomes relevant when x_2 is included. Obviously, one could set the screening threshold γ in SPCA so low that also x_1 would survive the screening, but in practice this means setting the threshold so low that basically all features are included, meaning that SPCA would in essence reduce to the standard PCA with the problems explained in Section 2.1.

4.2 Data Visualization

This section illustrates the use of ISPCA for visualization of high-dimensional real world data and shows how it compares to PCA and SPCA[1]. All the datasets involve a classification problem with the number of features ranging from about 1500 to 22000 and the number of training instances from about 50 to 2000. Our main interest are the "small n, large D" cases and most of the problems fall into this category, but we included also a few problems with moderately large n. See Table 1 and the associated text in the supplementary material for more information about the datasets.

---

[1] Unlike in Bair et al. (2006) who used a model dependent cross-validation scheme to chose the screening threshold for SPCA, we used a simpler strategy and computed the p-values for each feature based on a permutation test for the univariate scores and retained only features with $p < 0.001$.
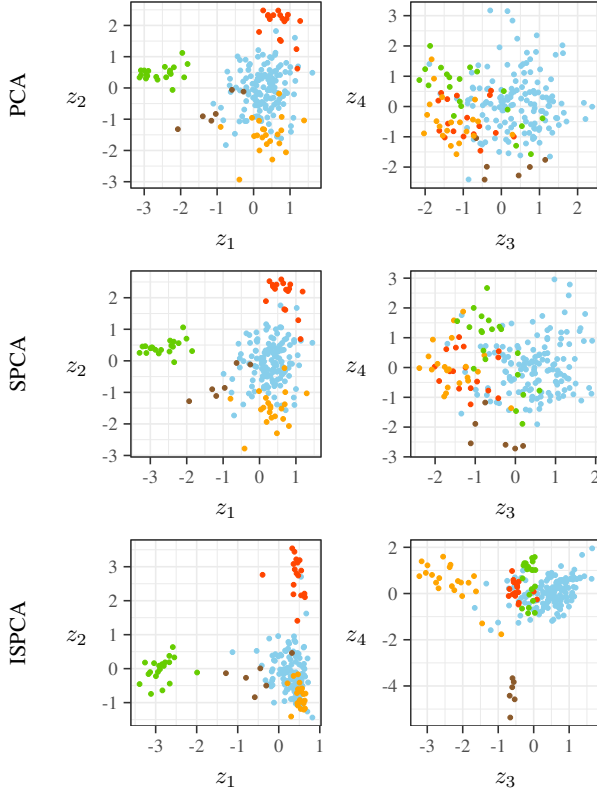
Figure 3: Visualization of Lung-5c cancer data ($n = 203, D = 3312$) using the first four latent features from PCA, SPCA and ISPCA. Different colours refer to the five different classes. Using only four features, ISPCA is able to separate the classes almost perfectly.

Figure 2 shows a representative set of the binary classification datasets visualized using the first two latent features obtained using the three methods. The benefit of supervision for visualization purposes is very clear: in many cases the two classes are considerably overlapping when visualized using the first two unsupervised PCs, but become fairly well separated when using either SPCA or ISPCA. By visual inspection, ISPCA seems to work clearly better than SPCA in at least one example (Basehock), slightly better in a few of the cases (PCMac, Arcene and Prostate) and slightly worse in one (Dexter).

Figure 3 shows a visualization of a dataset with five classes using the first four latent features of the three methods. Here PCA and SPCA perform very similarly; the first two latent features are informative for separating the red and green classes from the rest, but the remaining two features are only weakly informative and in this plot the classes are considerably overlapping. ISPCA on the other hand shows a substantial improvement; the method is able to find the third and fourth features so that also the orange and brown classes become well separated from the rest, and improves also the separation of the red and green class from the rest. By investigating which of the entries in the vectors $\mathbf{w}_1, \ldots, \mathbf{w}_4$

are nonzero we can get an idea about which of the features characterize the differences between the five classes (see Figure 6 in the supplementary material).

4.3 Predictive Model Construction

Finally, we shall consider how the different dimension reduction techniques perform when the extracted latent features are used for predictive model construction. In all cases we use the standard logistic regression model (in multiclass cases multinomial softmax regression) with the regularized horseshoe prior (Piironen and Vehtari, 2017c) for the regression coefficients. This prior shrinks the coefficients of irrelevant features heavily towards zero and softly regularizes the coefficients of relevant features. More details of the prior, implementation of the models and computation can be found from the supplementary material.

We tested the following dimension reduction methods:

- PCA: first K_{tot} unsupervised PCs (K_{tot} defined below).

- SPCA: supervised PCA; compute K_{tot} first PCs among those features with univariate score statistically significant ($p < 0.001$)

- SPCA-2: compute first $K_{tot}/2$ PCs among those features with univariate score statistically significant, and compute first $K_{tot}/2$ PCs using the rest of the features after subtracting the variation explained by the supervised components (see Sec. 3.3).

- ISPCA-naive: first K_{tot} ISPCs.

- ISPCA-small: first K ISPCs, decide K using the permutation test (Sec. 3.4).

- ISPCA: as ISPCA-small, but in addition compute $K_{tot} - K$ unsupervised components.

For the binary classification datasets we used $K_{tot} = 50$ and multiclass problems $K_{tot} = 20$. In addition we computed results also for ridge logistic regression and Lasso using the original features to get baseline results for the comparisons. The prediction accuracy was measured by splitting the data randomly into two parts, using one fifth as a test set, and then averaging the results over fifty such random splits.

Figure 4 shows the mean log predictive densities on test data for different methods on each dataset, the last plot denoting the average over all datasets (for classification accuracies, see Figure 5 in the supplementary material). The results show that ISPCA yields better results than PCA or SPCA for several datasets, but also loses to one of these in many cases, the overall result being very close with SPCA having a slight edge (see the last plot). Overall the dimension reduction techniques outperform Lasso by a clear margin, but the best method depends on data, which emphasizes that
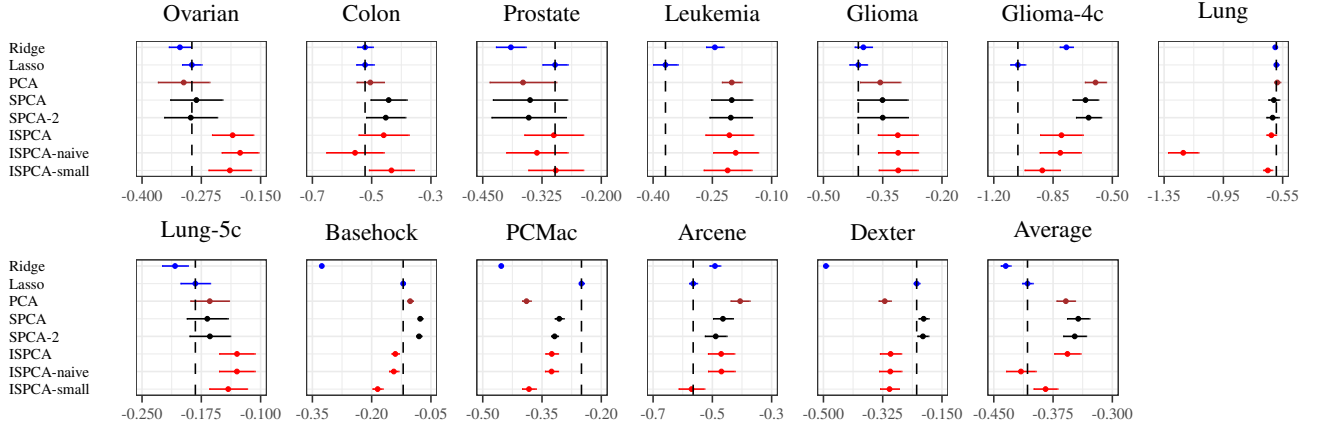
Figure 4: Mean log predictive densities (MLPD) on test data for the different methods on different datasets (larger is better). Horizontal bars denote the 95% intervals. The dashed vertical line denotes the performance estimate for the Lasso which was chosen as the baseline for the comparison. The last plot denotes the average over all the datasets.

no single method is optimal for every problem. The trend seems to be that ISPCA performs best on average for the microarray datasets (Ovarian – Lung-5c) whereas SPCA works better for the text classification datasets where the features are word counts (Basehock, PCMac, Dexter). This is an interesting pattern since it is somewhat at odds with the very good two dimensional feature representation of ISPCA which is better than for PCA and SPCA at least for Basehock and PCMac datasets (Figure 2). These results call for further investigation to better understand the successes and failures of each method.

The comparison of ISPCA to ISPCA-naive and ISPCA-small show that not using the permutation test can lead to inferior results due to overfitting (Colon, Prostate, Lung-2c), and that inclusion of the unsupervised features is basically never harmful, but can clearly improve the results in some cases (PCMac, Arcene). For SPCA the inclusion of unsupervised features does not appear to be crucial as the results for SPCA and SPCA-2 are practically the same for every dataset.

Computationally the three dimension reduction approaches are quite similar in the "small n, large D" realm, ISPCA being somewhat more expensive for multiclass problems and large n (see Table 2 in the supplementary). In these cases the number of supervised iterations is typically larger, which results in more permutation tests which are the most time consuming part of the method. However, even in these cases the computational bottleneck is still the fitting of the predictive model. Overall, although the considered methods cannot compete with Lasso in speed, the computation times are very affordable, considering that most problems allow the model to be fitted in a matter of seconds.

5 CONCLUSIONS

This paper has proposed a new supervised dimension reduction technique. Our experiments indicate that the proposed method is useful in many cases for visualizing high-dimensional labeled data as well as for reducing the dimensionality for predictive model construction. For visualization purposes, the proposed method appears to perform better than PCA or SPCA, which is due to the algorithm's greedy nature that tries to maximally load all the predictive power on the first few latent features. This makes the method useful for exploratory analysis. Regarding the predictive performance, although the method gave better results than the other methods on several problems, the experiments also demonstrated that it is not infallible and in some cases better results could be obtained by other means, such as the original SPCA. Based on the results it seems that none of the considered dimension reduction techniques is optimal for every problem, but in almost all cases at least one of them gave very good results (clearly better than Lasso or ridge), confirming that the dimension reduction approach is very viable alternative for (Bayesian) supervised learning in these problems and encourages further research and methodological development in this area.

As it stands, our pragmatic advice would be to use cross-validation for assessing the fit of the models obtained after the different computational shortcuts (such as SPCA and ISPCA) and to use the validation results to guide the model selection. We emphasize that it is advisable to validate also the dimension reduction process (that is, the dimension reduction is computed separately for each fold) to avoid any potential bias induced by conditioning the inference twice on the observed data.

The R implementations for the methods discussed in the paper are freely available at https://github.com/jpiironen/dimreduce.

References

Ahmed, T. and Bajwa, W. U. (2017). ExSIS: extended sure independence screening for ultrahigh-dimensional linear models. *arXiv:1708.06077*. 1

Bair, E., Hastie, T., Paul, D., and Tibshirani, R. (2006). Prediction by supervised principal components. *Journal of the American Statistical Association*, 101(473):119–137. 1, 2, 3, 5, 6

Bhadra, A., Datta, J., Polson, N. G., and Willard, B. (2017). Horseshoe regularization for feature subset selection. *arXiv:1702.07400*. 1

Candes, E. and Tao, T. (2007). The Dantzig selector: statistical estimation when p is much larger than n. *The Annals of Statistics*, 35(6):2313–2351. 1

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2010). The horseshoe estimator for sparse signals. *Biometrika*, 97(2):465–480. 1

Chang, C., Kundu, S., and Long, Q. (2016). Scalable Bayesian variable selection for structured high-dimensional data. *arXiv:1604.07264*. 1

Chen, Y. and Dunson, D. (2017). Modular bayesian screening for high-dimensional predictors. *arXiv:1703.09906*. 1

Dupuis, J. A. and Robert, C. P. (2003). Variable selection in qualitative models via an entropic explanatory power. *Journal of Statistical Planning and Inference*, 111(1-2):77–94. 5

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360. 1

Fan, J. and Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society. Series B (Methodological)*, 70(5):849–911. 1

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1). 11

George, E. I. and McCulloch, R. E. (1993). Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889. 1

Goutis, C. and Robert, C. P. (1998). Model choice in generalised linear models: A Bayesian approach via Kullback–Leibler projections. *Biometrika*, 85(1):29–37. 5

Mitchell, T. J. and Beauchamp, J. J. (1988). Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1036. 1

Mukhopadhyay, M. and Dutta, S. (2016). Bayesian variable selection for ultrahigh-dimensional sparse linear models. *arXiv:1609.06031*. 1

Neal, R. and Zhang, J. (2006). High dimensional classification with Bayesian neural networks and Dirichlet diffusion trees. In Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L. A., editors, *Feature Extraction, Foundations and Applications*, pages 265–296. Springer. 1

Paul, D., Bair, E., Hastie, T., and Tibshirani, R. (2008). "preconditioning" for feature selection and regression in high-dimensional problems. *The Annals of Statistics*, 36(4):1595–1618. 5

Piironen, J. and Vehtari, A. (2017a). Comparison of Bayesian predictive methods for model selection. *Statistics and Computing*, 27(3):711–735. 5

Piironen, J. and Vehtari, A. (2017b). On the hyperprior choice for the global shrinkage parameter in the horseshoe prior. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 905–913. PMLR. 1

Piironen, J. and Vehtari, A. (2017c). Sparsity information and regularization in the horseshoe and other shrinkage priors. *arXiv:1707.01694*. 1, 7, 10

Polson, N. G. and Scott, J. G. (2011). Shrink globally, act locally: sparse Bayesian regularization and prediction. In Bernardo, J. M., Bayarri, M. J., Berger, J. O., Dawid, A. P., Heckerman, D., Smith, A. F. M., and West, M., editors, *Bayesian statistics 9*, pages 501–538. Oxford University Press, Oxford. 1

Rockova, V. and George, E. (2014). EMVS: the EM approach to Bayesian variable selection. *Journal of the American Statistical Association*, 109(506):828–846. 1

Song, Q. and Liang, F. (2015). A split-and-merge Bayesian variable selection approach for ultrahigh dimensional regression. *Journal of the Royal Statistical Society. Series B (Methodological)*, 77:947–972. 1

Stan Development Team (2017). Stan modeling language users guide and reference manual, version 2.16.0. http://mc-stan.org. 11

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288. 1

Yu, S., Yu, K., Tresp, V., Kriegel, H.-P., and Wu, M. (2006). Supervised probabilistic principal component analysis. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 464–473, New York, NY, USA. ACM. 1

Zou, H. (2006). The adaptive Lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429. 1

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Methodological)*, 67(2):301–320. 1