Percentage of Variance Explained

Michael Kent

2015-09-21

Summary

This package extends the fastICA package (Hyvärinen, 1999) by providing a function which calculates a proxy for the variance of each of the separated signals, which otherwise have arbitrary variance. This proxy for variance is termed the *Percentage of Variance Explained* (PVE).

Details

1. Introduction

Hyvärinen and Oja (2000) describe Independent Component Analysis (ICA) as a method for separating non-Gaussian signals from Gaussian mixtures of signals, where independence between the signals is maximised. A popular algorithm used to perform ICA is the excellent fastICA algorithm by Hyvärinen (1999) and its R package by J. Marchini and C. Heaton. The algorithm implements the linear noise-free model of ICA which is shown in the following equation:

$X = SA$

The mixtures of signals are contained in the columns of X, while the mixing matrix is represented by A and separated signals in the columns of S. In the ICA model the signals have arbitrary variance as both S and A have to be estimated from the mixtures. Additionally, the signs, order of the signals and corresponding columns of A are also arbitrary.

In many applications, some signals are of more interest than others. Therefore a measure of importance is required to rank the signals (e.g.: by uncertainty or by a non-Gaussian measure). However, if the variance of the signals is of interest then the arbitrary variance of the signals poses a obstacle. To address the obstacle, this package proposes a proxy measure of variance, termed the *Percentage of Variance Explained* (PVE).

Associating a measure of variance to a signal is not uncommon. For example Westra et al. (2010) and Lotsch et al. (2003). However, it is not known how the specific measure of variance is calculated. Therefore the PVE calculation serves as a way of creating a proxy measure for the variance of signals. While there is currently no known formal proof for validating the PVE calculation, there is an application within Kent (2011).

Section 2 works through an example of the PVE calculation, while section 3 discusses the code for calculating the PVE of signals.

2. Worked Example of PVE

Load the package

```
devtools::load_all()
#> Loading pve
#> Loading required package: fastICA
```

Create artificial signals, each of length *n*. Signal *s1* is a sin wave while *s2* is a saw tooth function. *S* contains *s1* and *s2* in its columns. Plots of the signals are shown in figure 1 at end of example.

```
n <- 100
s1 <- sin(0:(n - 1) / 2)
s2 <- -(15 - abs(0:(n - 1) %% (2 * 15) - 15)) / 5
S <- cbind(s1, s2)
```

Create the mixing matrix. For this example, arbitrary values with range $-1 < i < 1$ are used. The mixtures of X are created from the product of S and A, and the column means are removed from the mixtures. Although the fastICA removes column means, the means are not returned by the function. As the means are needed by for the PVE calculation, the column means are rather removed and stored before performing ICA in *vcm*.

```
A <- matrix(c(-0.141, -0.293, -0.301, 0.603), nrow=2, ncol=2)
X <- S %*% A
vcm <- matrix(NA, nrow=1, ncol=ncol(X))
for (i in 1:ncol(X)){
  vcm[i] <- mean(X[, i])
  X[, i] <- X[, i] - vcm[i]
}
```

Next, perform ICA on X using the fastICA algorithm. For the PVE calculation to work, there must not be any dimension reduction at this point. Therefore the same number of signals as columns in X are separated from the mixtures.

```
ica_results <- fastICA(X, n.comp=ncol(X), alg.typ="parallel",
  method="C", fun="logcosh", verbose=TRUE, maxit=200, tol=1e-04,
  alpha=1)
#> Centering
#> Whitening
#> Symmetric FastICA using logcosh approx. to neg-entropy function
#> Iteration 1 tol=0.042198
#> Iteration 2 tol=0.000370
#> Iteration 3 tol=0.000001
```

Lastly, calculate the PVE for each of the signals and print the results. As the fastICA algorithm has a stochastic component, identical runs may produce slightly different results. This may effect the PVE values and the order that they are presented in. See the next section for details on the PVE calculation.

```
signal_pves <- calculate_pve_of_signals(X, ica_results$A,
  ica_results$S, vcm)
print(signal_pves)
#> [1] 86.84051 13.15949
```

The plots for the artificial signals, mixtures, and separated signals are shown below in figure 1 below.

```
par(mfcol=c(2, 3))
for (i in 1:ncol(S)){
  plot(S[, i], type="l", main=paste("Artificial Signal", i),
    xlab="", ylab="")
}
for (i in 1:ncol(X)){
  plot(X[, i], type="l", main=paste("Mixture", i), xlab="",
    ylab="")
}
for (i in 1:ncol(ica_results$S)){
  plot(ica_results$S[, i], type="l",
    main=paste("Separated Signal ", i, " PVE=",
    round(signal_pves[i], 2), "%",  sep=""), xlab="", ylab="")
}
```

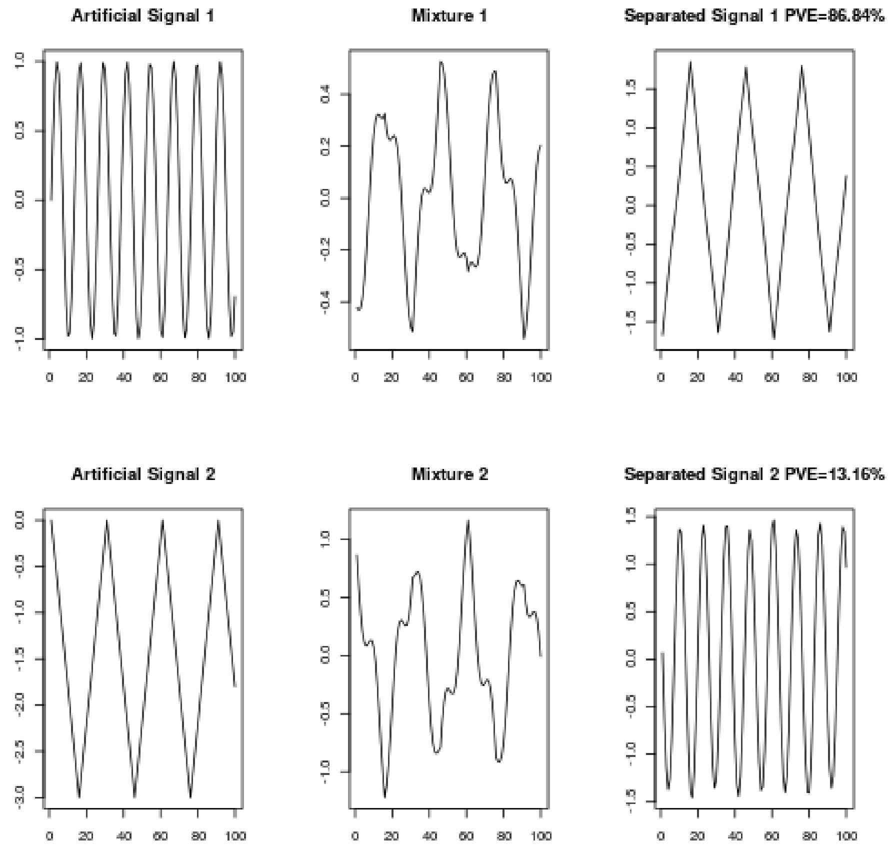

Figure 1: Artificial signals, mixtures, and separated signals in figure columns. Order, signs, and PVE of separated signals may change between identical runs.

3. PVE Calculation

The previous section worked through an example of the PVE calculation. This section discusses the function for calculating the PVE of signals and is presented below, though it is not a formal proof of the calculation.

```
calculate_pve_of_signals = function(X, A, S, vcm){
  original_var <- var(matrix(X))
  pve_list <- list()
  for (i in 1:ncol(S)){
    a <- matrix(A[i, ], nrow = 1, ncol = ncol(A))
    s <- matrix(S[ ,i], nrow = nrow(S), ncol = 1)
    var_s <- var(matrix((s %*% a))  + (vcm[i] / ncol(S)))
    pve_s <- var_s / original_var * 100
    pve_list <- c(pve_list, pve_s)
  }
  pve.list <- as.numeric(pve_list)
  return(pve.list)
}
```

The inputs to the function are: X Mixtures matrix containing mixtures of signals in columns. The matrix must have its column means removed (discussed below). A is the mixing matrix, S are the separated signals in columns of the matrix, and vcm is the vector containing the column means of X. Lastly the function returns a list containing the PVE for each signal.

The main part of the function is the line:

```
  var_s <- var(matrix((s %*% a))  + (vcm[i] / ncol(S)))
```

The line calculates the proxy variance for the i^{th} signal. The proxy is based on the following idea (var is the variance function):

Given the ICA model: $X = SA$

then $var(X) = var(SA)$

So too $var(SA) = \Sigma var(S_{n\times i}\ A_{i\times m})$ for $i \quad 1, 2, 3, ..., m$

Note that the variance is calculated on a one dimensional vector/matrix. The matrix is created by taking the corresponding two dimensional matrix and making into a one dimensional vector/matrix using the $matrix()$ function.

The next section of code adds back a portion of the column mean of X:

```
  + vcm[i] / ncol(S)
```

A portion of the total column mean is added to the signal to ensure that the total mean for AS remains equal to that of X in the ICA model. To do this a vector containing the column means (vcm) is used. This requires that before ICA is performed, the column means are removed from X as the fastICA algorithm removes them but does not return them.

Lastly, the following line takes the proxy for variance and scales it to represent a proportion of the input mixture variance. This becomes the PVE for the signal.

```
pve_s <- var_s / original_var * 100
```

References

A. Hyvärinen. Fast and Robust Fixed-Point Algorithms for Independent Component Analysis. *Neural Networks*, IEEE Transactions on, 10(3):626–634, 1999a.

A. Hyvärinen and E. Oja. Independent Component Analysis: Algorithms and Applications. *Neural networks*, 13(4):411–430, 2000.

M. Kent. The Value of Independent Component Analysis in Identifying Climate Processes. Master's thesis, University of Cape Town, 2011. URL: http://hdl.handle.net/11427/11457.

A. Lotsch, M. A. Friedl, and J. Pinzón. Spatio-Temporal Deconvolution of NDVI Image Sequences Using Independent Component Analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 41(12):2938–2942, 2003.

S. Westra, C. Brown, U. Lall, and A. Sharma. Interpreting Variability in Global SST Data Using Independent Component Analysis and Principal Component Analysis. *International Journal of Climatology*, 346(March 2009):333–346, 2010. doi: 10.1002/joc.1888.