Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Gene Expression

OXFORD

Gene Expression

Capturing patterns of variation unique to a specific dataset

Robin Tu [1,*], Alexander H. Foss [2] and Sihai Dave Zhao [1]

[1] Department of Statistics, University of Illinois at Urbana-Champaign, Champaign, 61820, USA
[2] Statistical Sciences, Sandia National Laboratories, Albuquerque, 87123, USA

* To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Capturing patterns of variation present in a dataset is important in exploratory data analysis and unsupervised learning. Contrastive dimension reduction methods, such as contrastive principal component analysis (cPCA), find patterns unique to a target dataset of interest by contrasting with a carefully chosen background dataset representing unwanted or uninteresting variation. However, such methods typically require a tuning parameter that governs the level of contrast, and it is unclear how to choose this parameter objectively. Furthermore, it is frequently of interest to contrast against multiple backgrounds, which is difficult to accomplish with existing methods.

Results: We propose unique component analysis (UCA), a tuning-free method that identifies low-dimensional representations of a target dataset relative to *one or more* comparison datasets. It is computationally efficient even with large numbers of features. We use publicly available protein expression and facial emotion data to demonstrate that UCA achieves similar results as cPCA with various tuning parameters, and UCA with multiple individual background datasets is superior to both cPCA with any single background data and cPCA with a pooled background data.

Availability: A free and open-source software implementation of the methodology, the UCA R package, is made available on GitHub. Code for all analyses presented in this article is also available via GitHub.

Contact: robintu2@illinois.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Capturing patterns of variation present in a dataset is an important step in exploratory data analysis and unsupervised learning. Popular methods include principal component analysis (PCA) [11], nonnegative matrix factorization [15], projection pursuit [10], and independent component analysis [8]. Frequently, however, many of the identified patterns may actually arise from systematic or technical variation, for example batch effects, that are not of substantive interest.

New approaches are necessary for capturing meaningful patterns of variation. A popular approach is to contrast a target dataset of interest with a carefully chosen background dataset that represents unwanted or uninteresting variation. Patterns of variation unique to the target, and not present in the background, are more likely to be substantively meaningful. For example, in Section 3.1, we analyze proteomics data from of normal and trisomic mice that have undergone Context Shock treatment and/or drug therapy. One goal is to identify patterns in normal mice relative to trisomic mice. However, the dominant modes of variation in the dataset arise from the Context Shock treatment and the trisomic gene, which are not of interest. Therefore, we contrast the data with a background dataset of proteomics data from Context Shock-treated and drug-treated trisomic mice. As a result, the remaining patterns of variation unique to the target dataset reveal features specific to normal mice.

This approach was first introduced by Abid et al. [1], who proposed contrastive principal components analysis (cPCA). While standard PCA identifies directions of variation that explain the most variation in the target dataset, cPCA seeks directions that explain more variation in the target than in the background. The most important patterns are those corresponding to the largest gap between the two datasets. Salloum and Kuo [19] later introduced cPCA++, which maximizes the ratio, rather than the difference, of the variance explained by the target and background. Boileau et al. [4]

described sparse cPCA, which seeks maxmially contrastive patterns of variation that can be characterized using a parsimonious set of features. Other types of contrastive implementations include latent models [20] and autoencoders [2].

There are two main issues with existing contrastive methods. First, a major disadvantage is that they cannot accommodate multiple background datasets. Using multiple backgrounds allows researchers to better hone in on the unique variation of interest by removing multiple types or sources of unwanted variation. For example, in the emotion analysis in Section 3.2, where we seek to uncover facial features characterizing the expression of "afraid", the dataset we use also contains background images from six other emotions. If we can simultaneously contrast the target data with multiple other background emotions, we will be able to identify more refined and distinctive patterns of variation characterizing "afraid". Naively applying existing methods by pooling the different emotions into a single background dataset is suboptimal, as variation in the pooled data may not be representative of variation in any of the individual datasets.

The second disadvantage of existing contrastive methods is that they typically require one or more tuning parameters. For example, cPCA requires the user to specify how much to penalize patterns that explain a large amount of variation in the background data. It is not clear in general how to choose these tuning parameters objectively.

We propose Unique Component Analysis (UCA), which addresses both of these issues. Not only can UCA contrast a target dataset against multiple backgrounds, but it also does not require any tuning parameters. UCA finds directions of variation that maximizes the explained variation in the target under a constraint on the amount of variation they can explain in each of the backgrounds. With a single background, UCA is equivalent to cPCA but with an automatically selected tuning parameter. We show that UCA achieves similar results as cPCA and cPCA++ with a single background and that it can outperform them when using multiple backgrounds. We also develop computationally scalable algorithms for application to experiments with large numbers of measured features.

The remainder of this article is organized as follows. In section 1.1, we present a brief overview of cPCA and cPCA++. Next, in section 2, we formally define UCA and mathematically detail our tuning-free, multibackground solution. Further, we present a computational algorithm to run UCA without the formation of covariance matrices. In section 3 we demonstrate UCA's superiority over cPCA and cPCA++ in two real world data examples involving mice protein expression and the Karolinska Directed Emotional Faces dataset. Additionally, we illustrate high-dimensional computational improvements by circumventing the formation of covariance matrices. Lastly, we conclude by reviewing the three main advantages UCA has over existing contrastive methods.

1.1 Background

1.1.1 Contrastive principal component analysis

We first briefly review cPCA [1]. Let A denoted the $p \times p$ sample covariance matrix constucted from target data $X_{n_x \times p}$ and B denote the $p \times p$ sample background covariance constructed from background data $Y_{n_y \times p}$. For some parameter λ, cPCA seeks eigenvectors $\hat{v}_\lambda \in \mathbb{R}^p$ that account for large variation in the target data Y and small variation in the background data X such that $\|\hat{v}_\lambda\|_2 = 1$. Specifically the first eigenvector is

$$\begin{aligned} \hat{v}_\lambda = \arg\max_{v \in \mathbb{R}^p} & \; v^T \left(A - \lambda B\right) v \\ \text{subject to } & \; v^\top v = 1 \end{aligned} \tag{1}$$

and, for example, the second eigenvector maximizes the quadratic form subject to be orthogonal to $\hat{v}_\lambda$.

For a given λ, $\hat{v}_\lambda$ can be interpreted as the direction that maximizes the variation in A without explaining much variation in B. The tuning parameter λ measures how much to penalize the background data covariance. When $\lambda = 0$, background variation is not considered, therefore cPCA reduces to PCA. As λ increases, the relative background variation becomes more dominant, causing $\hat{v}_\lambda$ to focus on directions which minimize background variation rather than maximizing the target. For very large values of λ, $\hat{v}_\lambda$ is equivalent to PCA after projecting the target data onto the nullspace of the background data. The authors of cPCA suggested that λ be chosen using spectral clustering via a parameter sweep of logarithmically spaced candidate values.

1.1.2 cPCA++

To eliminate the tuning parameter λ, Salloum and Kuo [19] proposed cPCA++, where they seek to maximize the ratio, rather than the difference, of the variance explained in the target to the variance explained in the background. Using similar notation as above where A denotes the target sample covariance matrix and B denotes the background sample covariance matrix, the first eigenvector of cPCA++ seeks to solve the generalized eigenvalue problem

$$\hat{v} = \arg\max_{v \in \mathbb{R}^p} \frac{v^T A v}{v^T B v}.$$

To compare to cPCA (1), we can also write the cPCA++ objective function in its primal form [12]:

$$\hat{v} = \arg\max_{v \in \mathbb{R}^p} v^T A v \text{ subject to } v^T B v = 1. \tag{2}$$

While maximizing the relative variability between A and B is tuning parameter free, the solution involves a matrix inversion, which may not be feasible in high-dimensional applications like genomics where the inverse may not exist. Furthermore, there is no clear path to extend this problem to multiple backgrounds.

2 Material and Methods

2.1 Unique Component Analysis

We introduce the Unique Component Analysis (UCA) framework, which combines ideas from both cPCA and cPCA++. First, we provide a reinterpretation of standard PCA. Applying PCA to a target dataset with covariance matrix A can be viewed as finding v that maximize the variance explained in the target, subject to explaining unit variance in a "background" dataset with covariance matrix I, where I is a $p \times p$ identity matrix. It is natural to use this white noise as a baseline because it contains no patterns of variation, as all features are uncorrelated. Thus any informative eigenvector should explain more variance in the target than in a white noise background (provided that features in the target dataset are scaled to unit variance).

This interpretation reveals some issues with the formulations of both cPCA and cPCA++. From , we can see that cPCA requires that its eigenvectors explain unit variance in a white noise background, but somewhat arbitrarily combines both A and B into a target matrix. Conversely, shows that cPCA++ uses A as the target but no longer requires its eigenvectors to explain unit variance in a white noise background. As a result, its eigenvectors are no longer comparable to those found by PCA and may actually explain less variance in the target than PCA eigenvectors.

To resolve these issues, we propose UCA, which joins the constraint of $v^T v = 1$ from PCA and cPCA with the constraint $v^T B v = 1$ from cPCA++. For a single background dataset, we propose to obtain the most important directions of variation by standardizing the features of the target and background to unit variance and then solving

$$\max_{v\in\mathbb{R}^p} \frac{v^T A v}{v^T v} \text{ subject to } \frac{v^T B v}{v^T v} \leq 1.$$

To make our procedure comparable to PCA, we constrain the eigenvectors to explain exactly unit variance in the white noise background. In addition, we require them to explain no more than unit variance in the background dataset. Notably, (2.1) does not require any tuning parameters.

Instead of directly solving (2.1), we instead solve the dual problem

$$\begin{aligned}
\max_{\lambda\geq 0} g(\lambda) &= \max_{\lambda\geq 0}\max_{v\in\mathbb{R}^P} \mathcal{L}(v,\lambda) \\
&= \max_{\lambda\geq 0}\max_{v\in\mathbb{R}^P} \left(\frac{v^T A v}{v^T v} - \lambda\left(\frac{v^T B v}{v^T v} - 1\right)\right) \\
&= \max_{\lambda\geq 0}\max_{v\in\mathbb{R}^P} \left(\frac{v^T (A-\lambda B) v}{v^T v} + \lambda\right) \\
&= \max_{\lambda\geq 0} (\lambda_{\max}(A-\lambda B) + \lambda), \qquad (3)
\end{aligned}$$

where λ is a Lagrange multiplier and $\lambda_{\max}(A-\lambda B)$ is the largest eigenvalue associated with $A-\lambda B$. This is convenient because dual functions are always concave [5]. We employ L-BFGS-B [6] to find the optimal λ and speed up convergence by calculating the gradient of $g(\lambda)$. If $\hat{v}_\lambda$ is the first eigenvector of $A-\lambda B$, normalized such $\hat{v}_\lambda^\top \hat{v}_\lambda = 1$, implicit differentiation can be used to show that

$$\frac{dg}{d\lambda} = -\hat{v}_\lambda^T B \hat{v}_\lambda + 1.$$

Finally, let $\hat{\lambda}$ denote the solution to the dual problem (3). Then we take the corresponding eigenvectors of the matrix $A-\hat{\lambda}B$ to be the components of our UCA algorithm. In this sense, UCA can be thought of as an automatically tuned version of cPCA with a contrastive parameter equal to $\hat{\lambda}$.

A major advantage of our UCA formulation is that we can accommodate multiple backgrounds. Let the A be the target $p\times p$ covariance matrix and $B_1,\ldots,B_m$ now be the m background $p\times p$ covariance matrices, constructed from an $n_y\times p$ dimensional target data matrix Y and corresponding $(n_{x_1}\times p),\ldots,(n_{x_m}\times p)$ dimensional background data matrices $X_1,\ldots,X_m$. We scale all features in all datasets to unit variance.

For each background $j=1,\ldots,m$, we add additional constraints $\frac{v^T B_j v}{v^T v} \leq 1$ to our optimization problem. Specifically, for multiple backgrounds, the primal problem becomes

$$\begin{aligned}
&\max_{v\in\mathbb{R}^p} \frac{v^T A v}{v^T v} \\
&\text{subject to } \frac{v^T B_1 v}{v^T v} \leq 1,\ldots,\frac{v^T B_m v}{v^T v} \leq 1.
\end{aligned} \qquad (4)$$

The corresponding dual function problem is

$$\begin{aligned}
&\max_{\lambda_1,\ldots,\lambda_m\geq 0} g(\lambda_1,\ldots,\lambda_m) \\
&= \max_{\lambda_1,\ldots,\lambda_m\geq 0}\max_{v\in\mathbb{R}^P} \left(\frac{v^T\left(A-\sum_{j=1}^m \lambda_j B_j\right)v}{v^T v} + \sum_{j=1}^m \lambda_j\right) \\
&= \max_{\lambda_1,\ldots,\lambda_m\geq 0} \left(\lambda_{\max}\left(A-\sum_{j=1}^m \lambda_j B_j\right) + \sum_{j=1}^m \lambda_j\right). \qquad (5)
\end{aligned}$$

We use coordinate descent to solve for $\hat{\lambda}_1,\ldots,\hat{\lambda}_m$, the solution to the dual problem (5). Then we take the corresponding eigenvectors of the matrix $A-\sum_j \hat{\lambda}_j B$ to be the components of our UCA algorithm.

2.2 Extension to High Dimensional Data:

Under the high-dimensional situation, where the number of variables p far exceeds the number of samples n, constructing the covariance matrix and using eigendecomposition to find the top eigenvectors becomes computationally expensive. To implement standard PCA, we can avoid creating and storing a large $p\times p$ covariance matrix by instead applying singular-value decomposition (SVD) to the data matrix.

Analogously, to extend UCA to high-dimensional data, we introduce the Product SVD method to exploit the structure of the contrastive problem so that we can use SVD and avoid constructing $p\times p$ matrices. For a single background, let the target data Y and the background data X be centered and scaled $n_y\times p$ and $n_x\times p$ matrices, respectively. If A and B are the corresponding covariance matrices, we can write $A-\lambda B$ as a product of a $p\times(n_y+n_x)$ dimensional left matrix, L, and a $(n_y+n_x)\times p$ dimensional right matrix, R, as seen in equation 6:

$$\begin{aligned}
C_\lambda &= A-\lambda B \\
&= \frac{1}{n_y}Y^T Y - \frac{\lambda}{n_x}X^T X \\
&= \underbrace{\left[\frac{1}{\sqrt{n_y}}Y^T, \frac{-\lambda}{\sqrt{n_x}}X^T\right]}_{L}\underbrace{\begin{bmatrix}\frac{1}{\sqrt{n_y}}Y\\ \frac{1}{\sqrt{n_x}}X\end{bmatrix}}_{R} \qquad (6)
\end{aligned}$$

With this formulation, we can follow the steps in Golub et al. [13], and find the singular values and vectors of C_λ using a sequence of SVD and QR decompositions operating on the left and right matrices. Since singular values and eigenvalues coincide in square matrices, we use the singular vectors, U, to find the largest eigenvalues by sorting the diagonal of $ULRU^T$. We describe the Product SVD Method in algorithm 1, which can directly replace the more computationally expensive eigendecomposition in high dimensions.

Algorithm 1: Product SVD Method to find largest eigenvalue of C_λ

Input: Centered background matrix X, centered target matrix Y, and λ;

1. Construct $L = \left[\frac{1}{\sqrt{n_y}}Y^T, -\frac{\lambda}{\sqrt{n_x}}X^T\right]$, $R = \begin{bmatrix}\frac{1}{\sqrt{n_y}}Y\\ \frac{1}{\sqrt{n_x}}X\end{bmatrix}$;
2. Find the SVD of the right matrix, $R = U_R S_R V_R^T$;
3. Find the QR decomposition of LU_R into $Q_{LU_R}R_{LU_R}$;
4. Find the SVD of the product of R_{LU_R} (step 3) and S_R (step 2), $R_{LU_R}S_R = EDF^T$;
5. The singular vectors of C_λ are the product of Q (step 3) and E (step 4), $U_{C_\lambda} = Q_{LU_R}E$;
6. Find the largest eigenvalues of C_λ by sorting the diagonal of D_{C_λ}, where $D_{C_\lambda} = U_{C_\lambda}C_\lambda U_{C_\lambda}^T$;

Output: $\lambda_{\max}(C_\lambda)$

Similarly, for multiple backgrounds, again let the A be the target $p\times p$ covariance matrix and $B_1,\ldots,B_m$ be the m background $p\times p$ covariance matrices constructed from a $n_y\times p$ dimensional Y data matrix and corresponding $(n_{x_1}\times p),\ldots,(n_{x_m}\times p)$ dimensional $X_1,\ldots X_m$ background data matrices.

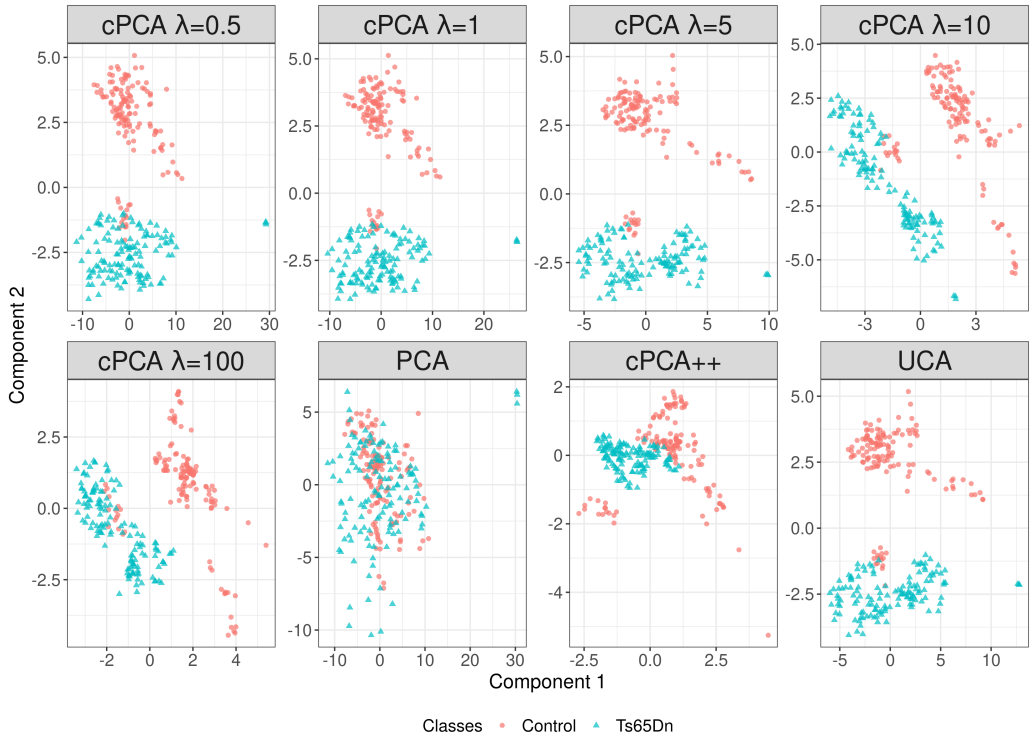


Fig. 1: Mouse Protein Expression: Normal and trisomic (Ts65Dn) mice separability of saline injected mice which were exposed to shock before getting environmental context (Shock Context). Both normal and trisomic mice receiving Shock Context treatment do not associate novel environments with adverse stimulus. Down Syndrome and normal mice were unseparable using PCA alone but are easily separable by cPCA at all values of the contrastive parameter λ. The mice are also easily separable by UCA. However, cPCA++ does not have similar separation performance compared to cPCA and UCA.

We can construct $C_{\boldsymbol{\lambda}} = A - \sum_{j=1}^{m} \lambda_j B_j$ analogously by appending the additional datasets to the left and right matrices:

$$\begin{aligned} C_{\boldsymbol{\lambda}} &= A - \sum_{j=1}^{m} \lambda_j B_j \\ &= \frac{1}{n_y} Y^T Y - \sum_{j=1}^{m} \frac{\lambda_j}{n_{x_j}} X_j^T X_j \\ &= \underbrace{\left[\frac{1}{\sqrt{n_y}} Y^T, \frac{-\lambda_1}{\sqrt{n_{x_1}}} X_1^T, \ldots, \frac{-\lambda_m}{\sqrt{n_{x_m}}} X_m^T \right]}_{L} \underbrace{\begin{bmatrix} \frac{1}{\sqrt{n_y}} Y \\ \frac{1}{\sqrt{n_{x_1}}} X_1 \\ \vdots \\ \frac{1}{\sqrt{n_{x_m}}} X_m \end{bmatrix}}_{R} \end{aligned} \tag{7}$$

We can apply this Product SVD method (algorithm 1) to solve the UCA dual problem.

The Product SVD method is advantageous in high-dimensions because it never explicitly operates on the entire $p \times p$ covariance matrix. Not only is our method more memory efficient, but our method is also computationally more efficient at finding the largest eigenvalue/eigenvector. By operating on the data matrix, our method scales with $n \times p$ bytes rather than p^2 bytes. Further, operating SVD and QR decomposition on either the left or right matrices is faster than directly operating eigendecomposition on the covariance matrix. In the single background scenario, λ only appears in L. Thus since, R doesn't change, we can pre-compute the SVD of R, and only update L when solving for λ_{max}. Similarly, in the multi-background scenario, rather than reconstructing $A^*_{(j)} = A - \sum_{i \neq j}^{m} \lambda_i B_i$ for each j in the original coordinate descent algorithm, this method allows us to only modify the j element of the left data matrix, L. The SVD of the right matrix, R, only needs to be computed once in our coordinate descent. Furthermore, at each step within the coordinate descent only step 3 and 4, the SVD and QR are computed, and are done on matrices with dimensions much smaller than $p \times p$.

All computations in this paper were done with R 3.6.3 [18] on an AMD Ryzen 1700X 3.7 gHz processor and 64GB 3000 mhz DDR4 RAM, Fedora 33 system.

2.3 Algorithm and software implementation

We have released a R implementation of UCA which is downloadable at `https://github.com/rtud2/Unique-Component-Analysis`. We have implemented both Product SVD on the data matrix and eigendecomposition on the contrastive covariance matrix and allow the background to take either a single background or a list of backgrounds. The GitHub repository also includes R markdown and datasets that reproduce most of the figures in this paper and in the Supplementary.

2.4 Data Availability

Datasets that have been used to evaluate UCA in this paper are publicly available from the authors of the original studies. The mouse proteomics data are available from the UC Irvine Machine Learning Repository `https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression` and the Karolinska Directed Emotional Faces (KDEF) are available from `https://www.kdef.se/`.

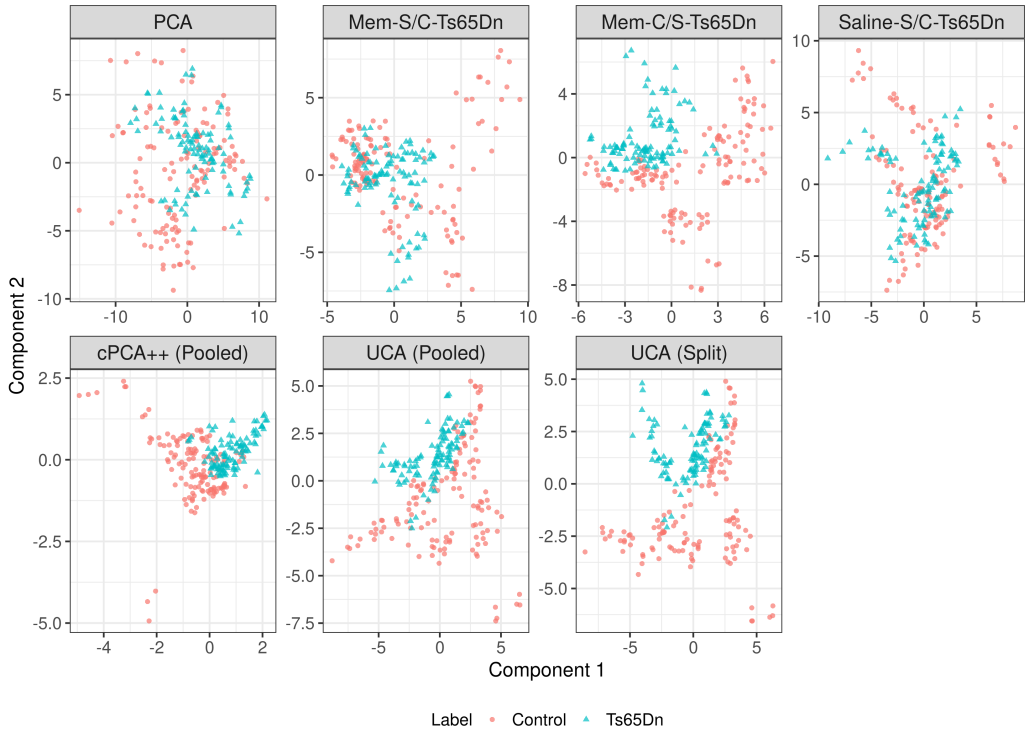


Fig. 2: Mouse Protein Expression: Separability of normal and trisomic Context Shocked mice injected with saline. UCA with a background dataset of Shock Context trisomic mice injected with memantine (Mem-S/C-Ts65Dn), UCA with a background dataset of Context Shock trisomic mice injected with memantine (Mem-C/S-Ts65Dn), and UCA with a background dataset of Shock Context trisomic mice injected with saline (Saline-S/C-Ts65Dn). Pooled: all background datasets (Mem-S/C-Ts65Dn, Mem-C/S-Ts65Dn, Saline-S/C-Ts65Dn) are pooled into one background. Split: UCA using multiple individual backgrounds separately.

3 Results

3.1 Discovering subgroups in protein expression data

We applied UCA to mouse proteomics data, which were also used by Abid et al. [1] to illustrate cPCA performance. The study measured levels of 76 proteins in 570 normal and 510 trisomic (Down Syndrome) mice receiving various combinations of learning therapies (Context Shock vs. Shock Context) and drug (memantine vs. saline) [3, 14, 1]. Normal mice exposed to Context Shock will first be exposed to novel context then expoed to shock, and learn to associate novel contexts with adverse stimulus; trisomic mice will not learn this association. Under the Shock Context treatment, mice are first exposed to shock than exposed to novel context, resulting in normal and trisomic mice not associating the environment with adverse stimulus. The goal of the experiment was to assess whether memantine improved learning ability in trisomic mice and to identify subsets of protein that may be involved in this process.

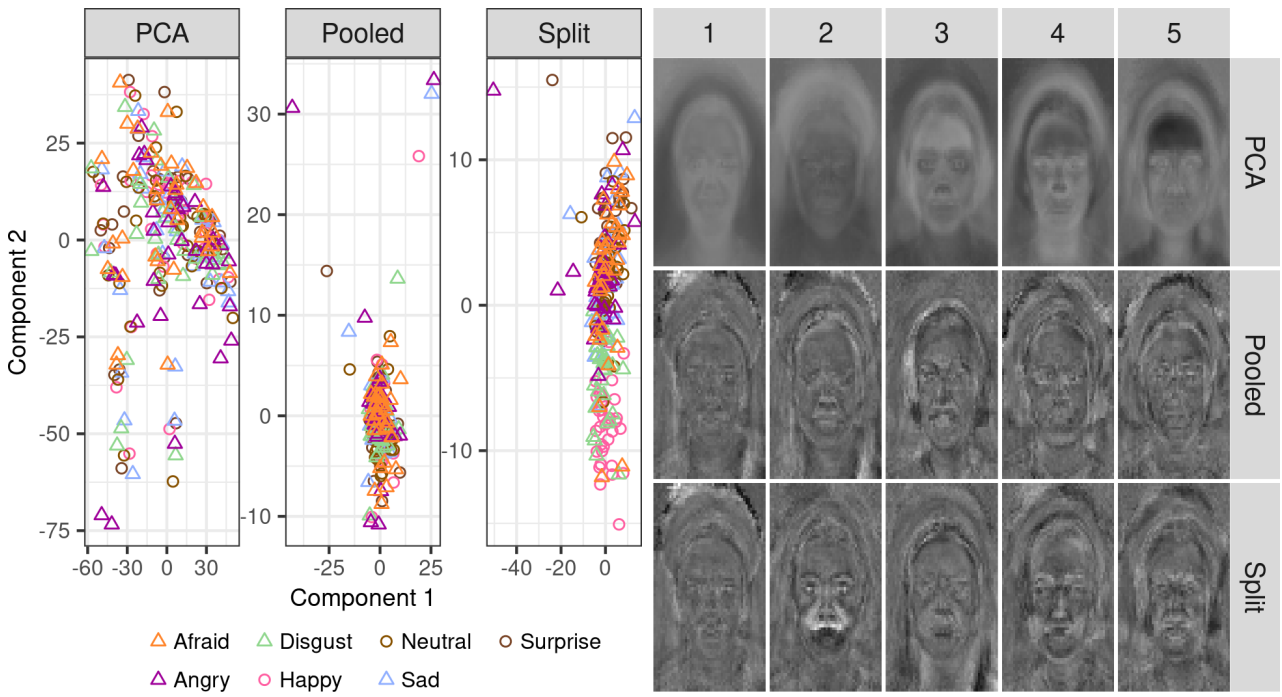
We first replicate the analysis by Abid et al. [1]. We want to extract patterns of variation in protein levels that can help discriminate normal from trisomic mice. Our target dataset consisted of protein data from normal and trisomic mice which received Shock Context and were given saline. However, natural variation, arising from factors such as age and gender, may dominate this dataset and obscure the variation of interest due to trisomy. To remove this natural variation, we contrasted the target data with a background dataset consisting of normal Context Shocked mice who had been given saline. As natural variation is likely present in both the target and the background, patterns that explaining variation in the target but not the background may likely to related to trisomy.

Figure 1 shows the data projected to the first two components identified by PCA, cPCA, cPCA++, and UCA. Normal and trisomic mice are not well-separated in the PCA results, showing that dominant variation in the target data indeed does not stem from trisomy. In contrast, the two mouse groups are much more clearly separated in the cPCA results. While this separation is noticeable for each of the tuning parameter values we tried, the actual projected data can vary considerably, and the optimal tuning parameter value remains unclear. cPCA++, which does not require specifying tuning parameter here because the number of samples exceeds the number of features, shows better separation than PCA but does not perform as well as cPCA. UCA performs as well as cPCA but without requiring a tuning parameter.

We next repeat the same analysis, but this time using Context Shocked mice given saline as our target dataset. Abid et al. [1] did not consider this analysis, where separating normal and trisomic Context Shocked mice proves to be a much more challenging problem. Figure 2 shows that normal and trisomic mice are again not well-separated by the first two components learned by PCA. Here we extract patterns of variation unique to normal mice and so applied UCA using three trisomic mouse datasets as backgrounds.

The results show that a single background dataset alone cannot separate normal and trisomic mice well. Pooling the three individual backgrounds together achieves slightly better separability compared to some of the individual backgrounds. However, naively constrasting multiple backgrounds simultaneously without pooling allows our proposed UCA to remove variability specific to each background and results in the best separability.

(a)



(b) (c)

Fig. 3: Analysis of KDEF faces. (a) Each emotion averaged over all female actors. (b) Faces projected onto the first two components of PCA, UCA with a pooled background, and UCA using multiple backgrounds. Shapes distinguish negative (triangle) and non-negative (circle) emotions. (c) Top 5 eigenfaces of female emotions produced by PCA, UCA with a pooled background, and UCA using multiple backgrounds.

3.2 Discovering eigenfaces of emotion

We further illustrate the advantages of contrasting multiple background datasets with UCA using the Karolinska Directed Emotional Faces (KDEF) [7], which captured images of seven emotions from five different angles from 70 amateur actors in either a practice or final session. Here, our target data includes all emotions from the final session. Figure 3a presents averaged images of each emotion calculated from all female actors in their final session.

We focus on uncovering variation unique to the "afraid" emotion using all final session pictures from every emotion from female actors. Since this target contains six other emotions, standard PCA will not provide variation unique to "afraid". Instead, we leverage images from the practice session to serve as background data. We construct six separate backgrounds, corresponding to the six other emotions, and contrast out their variation using UCA. For comparison, we also pooled these images together into a single background dataset and performed UCA using the pooled background.

Figure 3b projects the target data onto the first two directions of variation calculated using each method. The emotions are entirely intermixed when projected onto components produced by PCA and by UCA using a single pooled background. In contrast, the emotions are much more separable when using UCA with multiple separate backgrounds. In particular, "afraid" faces are very separate from the "disgust" and "happy" faces. This indicates that UCA with multiple backgrounds can identify patterns of variation that can distinguish "afraid" from the other emotions. Supplemental figure **??** plots the second component density to better show the differences in variability betwwen PCA, pooling, and splitting.

Figure 3c is a visualization of the top five directions of variation produced by each method as faces. These are called eigenfaces, and this visualization technique is useful for interpreting the top components as facial features [**?**]. PCA eigenfaces generally represent features common to all the emotions. UCA eigenfaces using a pooled background seem to highlight the eyebrows and upper lips, though it is difficult to discern. On the other hand, eigenfaces produced using UCA using multiple separate backgrounds highlight eyebrows, eyes, nostrils, and nasolabial folds. These are especially clear in the second eigenface and accord with intuition about which features would likely be most useful in distinguishing "afraid" from other emotions.

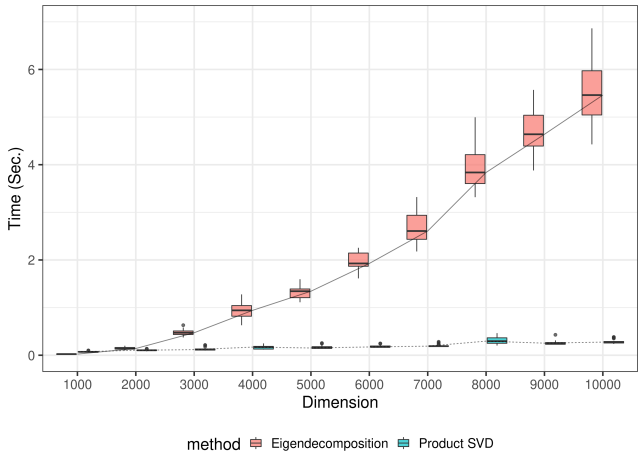


Fig. 4: UCA and cPCA implementation using Product SVD vs. eigendecomposition for high-dimensional data: 25 random $100 \times p$ target and background matrices are generated from a standard normal distribution and where p, the dimension varied from 1,000 to 10,000 in steps of 1,000. Box plots of time (in seconds) is plotted for both eigendecomposition and the product SVD method. For small p, there is a negligible difference between the methods. However, as dimension p increases, the Product SVD is significantly faster.

3.3 Product-SVD algorithm simulation

To demonstrate the speed improvements of the Product SVD method compared to the current fastest implementations of eigendecomposition in high-dimensions, we conduct a simulation study with 25 sample $100 \times p$ target and background data matrices generated from a standard normal distribution with p varying from 1.000 to 10,000 in steps of 1,000. To ensure a fair comparison, we leverage C++ in both implementations using a custom RcppArmadillo [9] function for the Product SVD method and the RSpectra package (0.16-0) [17] for the eigendecomposition method. RSpectra is a package designed for large-scale eigendecompositions based off the C++ Spectra library. We use the microbenchmark package [16] to ensure accurate timings. Our benchmark does not take into account the additional cost of forming the $p \times p$ covariance matrices, which would only exacerbate the difference between the two methods in real world applications.

Figure 4 shows box plots of time (in seconds) versus the dimension, p, colored by method, and summarizes the results of the simulation study. As the dimension p increases, the computational time of our Product SVD method increases much slower than the current eigendecomposition implementations. It should be mentioned that for $p < 1000$ the Product SVD method is slower due to overhead of additional computation on small matrices. In general, for low-dimensional settings where $n \geq p$, the Product SVD will be negligibly slower because of the additional QR, SVD, matrix products, and sort computations.

4 Discussion

In many data analytics settings, we are interested in removing uninteresting variation that contaminate the data of interest. Here we proposed UCA, a tuning parameter free contrastive learning method that simplifies cPCA while also substantially improving it by accommodating multiple background datasets. We demonstrate UCA's usefulness and superiority in several examples.

UCA is computationally fast and easily extensible to high-dimensional data because it does not require constructing nor storing a large covariance matrices (see Methods). Further, no additional post-hoc clustering method is necessary to choose the appropriate tuning parameter.

Like cPCA, the choice of background(s) still plays a pivotal role in the directions found by UCA. If two background datasets contain highly correlated information, the algorithm will weight them accordingly. For example, in the event of identical backgrounds, only one background will be considered.

While the goal of UCA is not always to find separability between groups, UCA emphasizes finding variation in the target data scarcely seen in the background data. Therefore switching the roles of the background and target data may not result in the same target data separation. For example, if our target data consists of all facial emotions in section 3.2, and our background consisted of negative emotions (Afraid, Angry, Disgust, Sad), we hope to see less variability in negative emotions and preserve variability in non-negative emotions (Happy, Surprise, Neutral). Conversely, using non-negative emotions as the background instead would preserve the variability in negative emotions, and decrease the variability in non-negative emotions. For maximal data seperation, we recommend using groups with larger variation as the background(s).

We have released the code for UCA as an R package, along with documentation and examples exhibited in this paper.

References

[1]Abubakar Abid, Martin J. Zhang, Vivek K. Bagaria, and James Zou. Exploring patterns enriched in a dataset with contrastive principal component analysis. *Nature Communications*, 9:2134, 05 2018.

[2]Abubakar Abid and James Zou. Contrastive variational autoencoder enhances salient features, 2019.

[3]Md. Mahiuddin Ahmed, A. Ranjitha Dhanasekaran, Aaron Block, Suhong Tong, Alberto C. S. Costa, Melissa Stasko, and Katheleen J. Gardiner. Protein dynamics associated with failed and rescued learning in the ts65dn mouse model of down syndrome. *PLOS ONE*, 10(3):1–25, 03 2015.

[4]Philippe Boileau, Nima S Hejazi, and Sandrine Dudoit. Exploring high-dimensional biological data with sparse contrastive principal component analysis. *Bioinformatics*, 36(11):3422–3430, 03 2020.

[5]Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6]Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

[7]Manuel G. Calvo and Daniel Lundqvist. Facial expressions of emotion (kdef): Identification under different display-duration conditions. *Behavior Research Methods*, 40(1):109–115, Feb 2008.

[8]Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287 – 314, 1994. Higher Order Statistics.

[9]Dirk Eddelbuettel and Conrad Sanderson. Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063, March 2014.

[10]J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C-23(9):881–890, 1974.

[11]Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[12]Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Eigenvalue and generalized eigenvalue problems: Tutorial. *arXiv preprint*

arXiv:1903.11240, 2019.

[13]Gene Golub, Knut Solna, and Paul Van Dooren. Computing the svd of a general matrix product/quotient. *SIAM Journal on Matrix Analysis and Applications*, 22(1):1–19, 2000.

[14]Clara Higuera, Katheleen J. Gardiner, and Krzysztof J. Cios. Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PLOS ONE*, 10(6):1–28, 06 2015.

[15]Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, Oct 1999.

[16]Olaf Mersmann. *microbenchmark: Accurate Timing Functions*, 2019. R package version 1.4-7.

[17]Yixuan Qiu and Jiali Mei. *RSpectra: Solvers for Large-Scale Eigenvalue and SVD Problems*, 2019. R package version 0.16-0.

[18]R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[19]Ronald Salloum and C.-C. Jay Kuo. Efficient image splicing localization via contrastive feature extraction. *CoRR*, abs/1901.07172, 2019.

[20]Kristen A Severson, Soumya Ghosh, and Kenney Ng. Unsupervised learning with contrastive latent variable models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4862–4869, 2019.